(2)

IDA PAPER P-2062

EVALUATION OF COMPUTER-AIDED SYSTEM DESIGN TOOLS FOR SDI BATTLE MANAGEMENT/C^3 ARCHITECTURE DEVELOPMENT

Dennis W. Fife, *Editor*

Kevin Campbell
John Chludzinski
Nelson Corcoran
Carlos Gonzalez
J. Bret Michael
Edgar Sibley
David Wheeler
Christine Youngblut

October 1987

Prepared for
Strategic Defense Initiative Organization (SDIO)

IDA

INSTITUTE FOR DEFENSE ANALYSES
1801 N. Beauregard Street, Alexandria, Virginia 22311

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

AD-A197 452

1a REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release - distribution unlimited
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) IDA Paper P-2062	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Institute for Defense Analyses 6b OFFICE SYMBOL IDA	7a NAME OF MONITORING ORGANIZATION DOD - Ida Management Office
6c ADDRESS (City, State, and Zip Code) 1801 N. Beauregard St. Alexandria, VA 22311	7b ADDRESS (City, State, and Zip Code) 1801 N. Beauregard Street Alexandria, Virginia 22311
8a NAME OF FUNDING/SPONSORING ORGANIZATION Strategic Defense Initiative Organization 8b OFFICE SYMBOL (If applicable) SDIO	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER MDA 903 84 C 0031
8c ADDRESS (City, State, and Zip Code) SDIO/SYS Room 1E149 Pentagon, Washington D.C. 20301-7100	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. T-R5-422 / WORK UNIT ACCESSION NO.
11 TITLE (Include Security Classification) Evaluation of Computer-Aided System Design Tools for SDI Battle Management/C3 Architecture Development. (U)	
12 PERSONAL AUTHOR(S) Dennis W. Fife, Kevin Campbell, John Chludzinski, Nelson Corcoran, Carlos Gonzalez, J. Bret Michael, Edgar Sibley, David Wheeler, Christine Youngblut.	
13a TYPE OF REPORT Final 13b TIME COVERED FROM _____ TO _____ 14 DATE OF REPORT (Year, Month, Day) 1987 October 1 15 PAGE COUNT 170	
16 SUPPLEMENTARY NOTATION	

17 COSATI CODES		
FIELD	GROUP	SUB-GROUP

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Strategic Defense Initiative; Battle Management/C3; TAGS; AUTO-G; DCDS; TEAMWORK; Software Through Pictures; computer software tools; SADMT; graphic computer-aided design; computer-aided software engineering (CASE); software specification; requirements analysis.

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This IDA Paper was prepared at the request of the Strategic Defense Initiative Organization. The paper documents the findings of an evaluation on the capabilities of certain computer software/computer-aided software engineering (CASE) tools to provide computer-aided graphic design of Battle Management/C3 for the SDIO. Each tool (of five selected on the basis of the best available at this time), was installed at IDA. After training by vendor tool staff, an IDA team, using a hands-on design exercise determined the merits of the tools for SDI application. A comparative summary of the tools is given relative to envisaged SDI requirements and an extensive questionnaire is answered for each.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ■ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Dr. Dennis W. Fife, IDA	22b TELEPHONE (Include Area Code) (703) 845-3512 22c OFFICE SYMBOL IDA/CSED

DD FORM 1473, 84 MAR

83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

IDA PAPER P-2062

EVALUATION OF COMPUTER-AIDED SYSTEM DESIGN TOOLS FOR SDI BATTLE MANAGEMENT/C^3 ARCHITECTURE DEVELOPMENT

Dennis W. Fife, *Editor*

Kevin Campbell
John Chludzinski
Nelson Corcoran
Carlos Gonzalez
J. Bret Michael
Edgar Sibley
David Wheeler
Christine Youngblut

October 1987

IDA

INSTITUTE FOR DEFENSE ANALYSES

Contract MDA 903 84 C 0031
Task T-R5-422

TABLE OF CONTENTS

TABLE OF FIGURES

APPENDIX A.

TAGS

AUTO-G

SOFTWARE THROUGH PICTURES

TEAMWORK

EXECUTIVE SUMMARY

Five computer software tools have been assessed for graphic computer-aided design of Battle Management/C^3 in the Strategic Defense Initiative (SDI) program. They are: TAGS from Teledyne Brown Engineering, AUTO-G from Advanced System Architectures (U. K.), DCDS from TRW and U.S. Army Strategic Defense Command, Teamwork from Cadre Technologies, and Software Through Pictures from Interactive Development Environments.

Each tool was installed at IDA and, after training by tool vendor staff, evaluated through a hands-on design exercise. This supplemented a review of documented capabilities, and provided more depth than is typical of software evaluations. Pertinent excerpts of this report were provided to the tool producers prior to publication and appropriate clarifications made from their comments.

Excluding DCDS, IDA evaluators found the tools readily useable for graphic design in the versions provided to IDA. DCDS lacks the graphic capabilities of the others, and only test versions of a major uncompleted upgrade were available.

The tools differ significantly in their current capabilities to represent real-time systems, support complex software development projects, and provide for design validation and simulation. In particular, only one, AUTO-G, now generates automatically the textual language representation (SDI Architecture Dataflow Modeling Technique, SADMT, formerly called SDI Architecture Process Description Language, SA/PDL) that will be used in testbed simulation by the Strategic Defense Initiative Organization (SDIO). AUTO-G has a demonstrable SADMT generator, with an updated package due for January 1988 delivery. TAGS is the only tool with its own demonstrable simulation capability. This generates an Ada simulation of a designed system. TAGS also directly provides configuration management and other design support needs. Hence, at this time, AUTO-G

and TAGS have greater capabilities and potential in areas most pertinent to SDI requirements. DCDS would be included also, except that its graphics interface is inferior to most leading CASE tools.

Though the five tools rank among the best available, none fully meets SDI requirements as perceived by IDA. To illustrate, the ten capabilities listed below are important requirements, but were available only for the tools named in parentheses[1]. (Where brevity is essential, Software Through Pictures is abbreviated as STP in this report.)

1. Provides effective graphics (TAGS, AUTO-G, STP, Teamwork);
2. Represents timing requirements (TAGS, AUTO-G, DCDS);
3. Represents process behavior formally (TAGS, AUTO-G, DCDS);
4. Represents hardware allocations explicitly (DCDS);
5. Generates SADMT representation for SDIO simulation testbed (AUTO-G);
6. Provides simulation for dynamic validation by designers (TAGS);
7. Provides requirements traceability (DCDS);
8. Provides version identification (TAGS, AUTO-G, Teamwork);
9. Supports configuration management (TAGS);
10. Supports user-tailored graphic icons and semantics (none).

A table following this summary highlights a comparative view of the tools in a popularized form. It is keyed by number to detailed questions about the tools, answered in Appendix B. Appendix B questions represent a complete statement of tool requirements investigated in this evaluation. Differences in the "smiles" given to each criterion are:

---

[1] The above list and the following table exclude capabilities planned for future release or available through auxiliary packages. As examples, TAGS is planning to have SADMT in 1988, and through a separate package, RVTS, now provides requirements tracing. Version identification for STP is provided through an auxiliary package. Teamwork will have configuration management in 1988, as will STP via a separate package.

Graphics and Editing. Panning and picking in AUTO-G are relatively difficult, though scroll bars are being added in a future version. DCDS is much worse on panning and has no high resolution graphics to support large, fully labeled diagrams.

Design semantics and support. AUTO-G provides the SADMT language. AUTO-G, TAGS, and DCDS provide similar depth of formal specification capability. STP and Teamwork lack formal specification of processes and timing requirements.

Team design support. TAGS provides configuration management. TAGS, STP, and Teamwork provide locks for concurrent work by multiple designers on a single design project or database. AUTO-G has well-developed versioning. DCDS has none of these.

Documentation and output. TAGS provides a variety of standard database reports for design support. The others provide the basic means of output and database query, without significant distinctions. Tool documentation is not exceptional for any, and is outdated for DCDS.

Static diagnostics. AUTO-G provides the best form of error reports. AUTO-G and TAGS, because of their depth of formal specification, must check syntax and semantics more extensively. STP has no diagnostics for state transition tables[2].

Simulation. Only TAGS now offers a self-standing simulation capability.

Adaptability. DCDS alone has a database extension feature that is integrated within its textual specification language. All the tool vendors either now publish or will provide users with the information needed to access their database. None directly supports user modification of design icons or their semantics associated with the design database.

The IDA exercise was limited in important respects. IDA has not exercised the tools on a large team project, and so cannot predict their stability, reliability, or productivity

---

2 For the July 1987 version provided to IDA; since added to current STP release.

under the project stresses to be expected for SDI. To reach a final conclusion about their relative merits for SDI use, all of them should be applied in SDI development over the next six months. Extensions and refinements delivered by their developers also should be monitored, since improvements now under development will change their comparative standing.

SUMMARY OF FINDINGS

CAPABILITY (Pertinent questions in Appendix B)

Graphics and editing (1.5-1.6, 2.1-2.17)

TAGS	AUTO-G	DCDS	STP	TEAMWORK
☺☺☺	☺☺		☺☺☺	☺☺☺

Design semantics and support (3.1-3.11, 5.1-5.6)

TAGS	AUTO-G	DCDS	STP	TEAMWORK
☺☺☺	☺☺☺☺	☺☺☺	☺	☺

Team design support (1.1-1.4, 7.1-7.9)

TAGS	AUTO-G	DCDS	STP	TEAMWORK
☺☺☺	☺☺		☺☺☺	☺☺☺

Documentation and output (9.1-9.6)

TAGS	AUTO-G	DCDS	STP	TEAMWORK
☺☺☺	☺☺	☺☺	☺☺	☺☺

Static diagnostics (4.1-4.3)

TAGS	AUTO-G	DCDS	STP	TEAMWORK
☺☺☺	☺☺☺☺	☺☺	☺	☺☺

Simulation (8.1-8.2)

TAGS	AUTO-G	DCDS	STP	TEAMWORK
☺☺				

Adaptability (6.1-6.5)

TAGS	AUTO-G	DCDS	STP	TEAMWORK
☺	☺	☺☺	☺☺	☺☺

1.0 PURPOSE AND SCOPE

This report summarizes capabilities of computer software tools that provide a computer-aided, graphic means of describing systems. The tools examined are often categorized as computer-aided software engineering (CASE) tools. They apply to decomposing and describing system functions which ultimately may be implemented by hardware or software. The target system's operating environment must be defined also, at least in terms of input and output interfaces. The tools do not force a particular demarcation between the environment and the system under design. To a degree, they serve also as general system engineering tools.

IDA staff and consultants evaluated the tools by designing an elevator control system as described in Appendix C. This report captures answers to a set of questions about tool capabilities as available and used by IDA in August, 1987. The overall conclusions are meant to highlight use of the tools by design teams developing a complex, real-time control system.

Readers are assumed to have background in current software design methods and supporting computer-aided tools, so no tutorial is given. Pertinent background references are listed. An overview in Appendix A introduces each tool. Brief notes about near-future enhancements are included also, where the tool developers provided definite information. The next three sections describe the evaluation effort further, and provide summary conclusions on the usability, strengths, and weaknesses of the five tools for the Strategic Defense Initiative.

2.0 EVALUATION APPROACH

Prior IDA work has assisted this effort with pertinent background. In January, 1987, for example, IDA sponsored a Tools Fair at which 15 vendors of CASE tools provided briefings and demonstrations for contractors and SDIO officials [Heystek]. Except for Software Through Pictures, all of the tools examined here were presented at the Tools Fair. IDA subsequently investigated the design formalisms underlying selected tools [Chludzinski], including TAGS, AUTO-G, and DCDS. IDA also has developed the SADMT language [Linn] as a means to uniformly represent complex systems and implement an objective and maintainable testbed for simulation and evaluation.

The objective of the effort reported here was to provide a timely assessment of the usability of selected tools that were under consideration for near-term use in developing SDI BM/C^3 proposals. Tools were selected that offer the quality and resolution of workstation graphics, and that are off-the-shelf, commercial products. DCDS was included because of its past history of use in defense projects. Available effort for the design exercise was limited, and other tools that also might qualify were not evaluated. Most of the choices made were subsequently identified as leading candidates by SDIO's BM/C^3 contractors.

The tools were installed at IDA, beginning with TAGS in June, 1987, with the last one, DCDS for the IBM PC-AT, installed in August, 1987. An exercise team of two or more IDA research staff or consultants was designated for each tool. Training was conducted by tool vendor personnel after each tool's installation. The teams were given the design requirements, Appendix C, and the questionnaire before beginning to use the tools. Documentation of this report began in mid-August.

The tools were used in the manner of a system designer as the online user, rather than the designer working offline with a tool/workstation operator. Whether a given tool, when used directly by a designer, improves overall designer productivity, may be

debatable. The IDA evaluators advocate the pro position, but IDA did not attempt any productivity measurements. Appendix A provides an overview of each tool, with selected diagrams or display prints to give the flavor of its graphics. Appendix B presents the answers to IDA questions in a side-by-side form.

3.0 ANALYSIS

The following paragraphs discuss the tools relative to broadly stated requirements perceived by IDA. Specific capabilities investigated are identified in the Executive Summary and detailed in the questions posed and answered in Appendix B.

Dozens of CASE tools are marketed today, and commercial competition is strong. Thus, off-the-shelf tool capabilities should be expected to change rapidly for the better. In most cases, the IDA evaluation identified bugs or badly needed extensions, and some developers attempted to fix these promptly. Most of the tool developers identified major improvements planned and already underway for 1988.

3.1 GRAPHICS AND USER INTERFACE

Highly productive design with a graphic tool begins from an effective and reliable user interface, which includes displays, menus, keying sequences, mouse usage, and other elements dominating the designer-system interaction. Except for DCDS, all of the tools offer well-developed and useable graphics for decomposing and describing systems. System description is done in terms of a "design language" embodied in the graphic icons (or text equivalents) and other tool features. The designer's composition of a description, by selection, labeling, and interconnection of icons, creates an internal database representation from which other tool functions and designer actions proceed.

The languages of Teamwork and Software Through Pictures are similar and based on well-known and widely used data flow and control flow diagramming methods. TAGS, DCDS, and AUTO-G have distinct and unique design languages, with various similarities to common notations such as block diagrams and flow charts. The next subsection on System Representation discusses the semantics of the design languages, excluding the graphics and user interface.

DCDS was used in beta-test versions of its Ada implementation for VAX and IBM PC/AT hardware. The beta-test software has many limitations or bugs, which are to be

overcome in the "production" version for the VAX, available in November, 1987. DCDS is severely limited in graphics capability, compared to the other tools, but this may improve with a proposed upgrade to Sun workstations, planned for 1988 release. The observations in this subsection do not include DCDS, because of its limitations.

None of the tools/workstations was found to be extremely easy to use, though IDA evaluators experienced least difficulty in using Teamwork. All the tools have some non-obvious, error-prone, or otherwise bothersome features relative to common user actions or situations. Users adjust to these in a short time, but initially find them frustrating.

TAGS, Software Through Pictures, and Teamwork require the user to choose the position of most, if not all, of the graphic icons and labels in the diagrams representing a system. IDA users experienced distinct graphic difficulties arising from the ability to position icons at will. A common difficulty was that some aspect of the internal database description became hidden or invisible on the displayed image. This arises from user actions that either overlay one object on another, or fail to delete all parts of an object. In these cases, diagnostics from analyzing the database content may note errors that cannot be identified from the visible diagram.

The AUTO-G software, on the other hand, automatically positions icons and text, and redraws the display after each editing action, avoiding any hidden or overlapping elements. Occasionally, this produces an annoyingly long response time. Also, it tends to spread a diagram widely, which may increase the trouble of panning across a large diagram. A repositioning feature allows a user to easily "squeeze" a diagram, but this doesn't eliminate the need for design discipline to create only small diagrams. A user may suppress or "hide" chosen parts of the picture also, to concentrate on other parts.

TAGS is the only system which encourages small diagrams by providing only a single page drawing or text entry window for its display. The designer may use a succession of pages, with off-page connectors to join flow lines for diagrams. Panning, scrolling, and zooming are unnecessary. The other tools offer a window to a very large drawing area for each diagram. So, user discipline is needed to avoid very large diagrams and extensive panning or zooming for viewing them. An AUTO-G user can limit diagram size and complexity through its modularity features (templates and separate documents) as well as its hiding feature. Teamwork and Software Through Pictures users would follow

some rule such as "only 3 to 7 bubbles per diagram" and the usual top-down decomposition technique. However, scrolling of a single diagram or table is easy with either of them.

3.2 SYSTEM REPRESENTATIONS

A crucial issue for SDI is the extent to which a tool provides for specifying all functions and attributes of a BM/C^3 system in a high level, formal language. IDA considers a formal language, rather than free text comments, as a paramount need because of the importance of testing and validation for the SDI. Elements of a formal specification can be cross-checked by computer, automatically and reliably, for consistency and completeness. Text annotations and comments, though valuable for human review, only support laborious and error-prone human effort to find errors. It is difficult to describe a minimum set of functions and attributes that should be formally specifiable for SDI, without presenting a mathematical or formalized notation. To avoid the latter, two alternatives are used below. The first is a conventional, informal description of requirements for real-time, database, and artificial intelligence types of systems. The second alternative considers the system characteristics or semantics expressible in the SADMT language, which SDIO requires as a deliverable representation from BM/C^3 contractors.

Informal View

All of the tools provide a way to decompose a system, hierarchically, into operating components and subcomponents. All provide one or more methods of describing information transfer from one component to another. All provide for describing some forms of control, such as the activation of one component or activity by another. All except DCDS support a purely functional or logical decomposition approach, as opposed to a physical one, if the designer so chooses. DCDS is the only tool that provides explicit allocation of function to physical components, but requires this choice earlier than may be appropriate for some projects.

Only TAGS, AUTO-G, and DCDS also provide for formally specifying absolute and relative time constraints or deadlines. This is crucial for real-time systems.

TAGS, AUTO-G, and DCDS also provide for formally specifying the behavior of system activities or processes at a detailed and algorithmic level. They include diagnostics at the algorithmic level.

Teamwork and Software Through Pictures provide state transition diagrams or tables, and related or alternative diagrams, such as decision tables, for specifying system control. In theory, such diagrams or tables suffice for describing any computable function, ignoring timing. Thus, control may be completely and formally specified with these two tools. But, formal linking of control flow with data flow is missing. Data flows and control flows may be viewed together on a composite flow display, but a data flow cannot appear in a control specification table or diagram. So, the tools do not include a formal basis (e. g., a decision table) for specifying data flow values in relation to control flow values. They cannot serve by themselves to formally specify the full algorithmic behavior of a system.

The common means of specifying process behavior with these two tools is a user-created form of structured text, say, PDL or Ada code. Both Software Through Pictures and Teamwork provide simple, though somewhat different templates for the text entry. The templates are produced automatically with variable and type declarations corresponding to the input and output data flows defined beforehand by the user's data dictionary entries. (An error in doing so for Ada enumerated data types is noted in the questionnaire for Software Through Pictures.) TAGS, AUTO-G, and DCDS also produce compilable data declarations from the formalized entries in their data dictionaries.

Teamwork and Software Through Pictures permit any textual description a designer wishes to make of a process, as do other tools. These are not checked by either tool's diagnostic analysis (because formal process descriptions are not provided within the tool) nor otherwise processed within the tool itself. Of course, if the designer enters a formal language text, such as Ada code or SADMT, then a separate compiler or other software could provide syntax checking or other support.

Database applications additionally need convenient specification of the database schema, in a form akin to the schema languages of pertinent database management software (DBMS). DCDS fulfills this requirement through a textual schema-like language. Software Through Pictures and Teamwork provide the capability to specify entity-

relationship-attribute (ERA) diagrams, often used to portray database schema. Software Through Pictures provides a command to produce a hierarchical text listing corresponding to an ERA diagram; Teamwork does not. The text form helps to prepare a database definition for prototyping or simulation on a DBMS package. Neither TAGS nor AUTO-G provides a means of relating sets of data elements in the form of a schema. All the tools would require some user-written text processing to manipulate their data dictionary content, as dumped, into the form of a desired schema language.

None of the tools appears to have techniques especially tailored to specifying knowledge-based software.

SADMT Semantics

SADMT requires a system's representation as a hierarchy or tree of asynchronously operating, intercommunicating processes (computations or any other behavior). Besides a process hierarchy, typed input and output ports, data flow interconnections, and Ada code for process behavior description, the SADMT language provides certain specification capabilities that are separable from the designed system specification. These are: port structure and type constraints; message constraints; event ordering constraints; and allocation of processes to components of stated hardware configurations (resources).

With regard to SADMT semantics, AUTO-G, TAGS, and DCDS could provide most, but not all, of the capabilities involved from their present design languages. The generation of a process tree from the internal database should be straightforward, though not equally easy for each. AUTO-G, TAGS, and DCDS provide for timing and assertions, but the specifications are embedded within the designed system specification. Depending on a tool's internal representation, some difficulty could arise in extracting and separating these to generate valid SADMT.

Neither AUTO-G nor TAGS provides a special means to explicitly describe hardware configuration and resource allocations. DCDS supports explicit definition of hardware configuration and associated functional allocations for a given system design.

Teamwork and Software Through Pictures could readily provide the process hierarchy, which is well-matched to their decomposition approach. Both lack timing, as already noted, as well as assertions and hardware configuration/allocation.

Again, all of the tools allow anything to be stated as a textual annotation, but because comments are not processed, even for static validation, they lack forceful effect. Automated diagnostics are the sine qua non of any specification capability. Nevertheless, it is beneficial to be able to create such annotations as a designer recognizes a technical issue or trade-off, or makes a decision that should be recorded. It is also useful to be able to apply other tool capabilities to such text annotations.

3.3 DATA COMMUNICATIONS REPRESENTATION

Data communications between physically separate BM/C^3 elements is as crucial to SDI as communication between software modules. Effectiveness in representing data communications functions is therefore an important criterion for SDI system and software engineering tools.

Teamwork and Software Through Pictures have no explicit mechanisms for specifying data communication interchange. All communication is shown by data flows and data stores (where buffering and delay occurs). However, representing some aspects of practical communication runs counter to the design philosophy inherent in data flow diagrams (DFDs). For instance, depicting physical elements and implementation properties is discouraged in DFD dogma. Moreover, multiple objects of the same kind typically would not be pictured as separate processes on a DFD. Thus, point-to-point communication between numerous elements may be difficult to describe. Showing separate data flows between every possible communicating party also leads to very complex diagrams, so unbuffered broadcast communication is difficult to represent with these tools. And, once again, the lack of a formal means to tie data and control flow together limits the capability to specify communication protocols.

TAGS and AUTO-G have well developed semantics applicable to data communications. TAGS semantics implies an underlying (unspecified) transmission medium without buffering or coordination. The system designer works with synchronous communication primitives (call, listen, send, and receive), and must fully specify the consequences (lost messages, retransmission, etc.) of a non-responding recipient. AUTO-G assumes an underlying asynchronous, buffered transmission medium (messages may be sent without explicit pre-coordination, but unexpected messages may be lost). For both of

these tools, a designer would explicitly represent any physical communication facility that operated in a different fashion.

DCDS provides both synchronous and asynchronous communication primitives and means to define the performance of an external communication network.

3.4 SADMT TEXT GENERATION

As mentioned already, AUTO-G is the only tool that is delivered with an automatic generator of the SADMT language, to support testbed simulation of a designed system. A working prototype of this generator was included in the version supplied to IDA. Teledyne Brown has stated their intent to furnish an SADMT generator with TAGS in 1988.

3.5 DESIGN SUPPORT

In addition to the basic graphics and design language, a highly effective tool provides features that expedite design work. This area addresses general productivity from an individual designer's viewpoint, excluding graphics and user interface as such, but recognizing practical aspects of design work with a given tool.

None of the tools comes with a designer's manual on how to carry out whatever design strategies the tools are built to support. Only AUTO-G had practical tutorial examples; others had very simple and limited examples. The IDA evaluators gained a distinct impression, from training and support contacts, that some tool producers have few personnel with strong experience in design.

Inexperience fosters impractical notions about design, such as: designers should (or will) create entirely satisfactory specifications, start to finish, without revising or starting over. None of the tools offers automatic aid for making revisions, such as propagating revisions from one change point to all affected definitions and diagrams. The views and both user-defined and automatic version identifiers of AUTO-G, however, are very helpful in tracking design updates and doing major cut-and-paste revisions. TAGS has time-date stamping available to track updates, but its version identifiers apply only when formal configuration management discipline is invoked. Teamwork provides automatic version identifiers and time-date stamping. Software Through Pictures provides version control through an external package. DCDS, perhaps because of a bug, usually will not save

incomplete and therefore syntactically invalid work. Its database translations for moving from one phase to another pose an added penalty for backing up to a prior design stage.

All of the tools constrain the order of going about the specification effort. DCDS is least flexible, imposing a highly ordered, start to finish method, which is enforced by its required succession of database translations, as well as the above mentioned bug. Teamwork and Software Through Pictures, by separating diagramming and text tools, encourage the traditional DFD approach of going level by level, creating first a correct diagram, then the data dictionary entries for it. AUTO-G is quite flexible. Naturally enough, it requires process behavior to be developed top-down, but the designer can easily move among many processes (if on the same diagram) to develop different aspects or stages of the design at will. TAGS supports similar freedom, but by its separate tools and specification forms (e.g., IORTD, IOPT, etc.), encourages a designer to work with one form at a time.

Inconsistent interpretations of published method or specification semantics was found on two tools. In TAGS, the Diagnostic Analyzer and the simulation generator embody different interpretations of the IORL (graphic) language, in some instances, than the reference manual of the language. IDA believes these are implementation discrepancies that will be corrected in time. Software Through Pictures allows multiple processes in a context diagram (a clear violation of published DFD dogma). When these are decomposed, irresolvable numbering conflicts arise between levels, so this permissiveness may lead to trouble. The tool does not allow external interfaces or entities to be introduced at lower levels, consistent with dogma. All external interfaces must be defined at the outset on the context diagram. For a system as complex as SDI, this forces the top level context diagram to be extremely cluttered and hard to follow. The interface complexity of a BM/C^3 system will demand a means of "hiding" the multitude of interfaces, and then progressively exposing them at successively lower levels of detail. IDA believes that this could easily be permitted by modifying the diagnostic checker of the tool.

3.6 DESIGN VALIDATION AND SIMULATION

Designers of complex systems need both static and dynamic validation aids from a tool. Static validation ensures completeness and integrity of a design as a whole, so that the parts match as defined and can be used for further automated processing, such as PDL

or code generation. Dynamic testing or simulation permits the designer to review the expected behavior of the system in operation. Dynamic testing also should be supported even when a design is incomplete, so long as some minimum specification has been done. The TAGS simulation capability, which supports simulation at any IORTD level, demonstrates this, as do other specification tools not evaluated here [Zave]. (This is provided in SADMT, for example, through the parent or non-leaf processes.)

The static validation provided by the tools basically involves completeness checks of individual diagrams, consistency checks between diagrams at different levels (balancing), and cross checking diagrams and associated tables where appropriate. Software Through Pictures had a diagnostic limitation in the version IDA used, in that no checking of control tables was done. This has been added in the current version. A noteworthy difference is how errors are presented. AUTO-G, for example, places error messages on the diagram, proximate to the error, which makes diagnosis especially convenient. TAGS provides a separate error listing, but keyed to the diagram by reference numbers automatically added by TAGS. Software Through Pictures provides a separate listing identifying erroneous elements by the labels, if any, on a data flow diagram. Unlabeled or hidden ones may be extremely difficult to locate. Appendix B notes that for one type of diagram, error messages refer to "lines", which are not elements of the diagram. Teamwork provides a separate error listing also, identifying errors by the context in which they appear on the diagram. IDA had no difficulties in locating errors using Teamwork's listing.

At this time, TAGS is the only tool that provides for simulation of a designed system (not exercised by IDA). The simulation involves Ada code generated on the Apollo workstation from a design database that passes the static diagnostics. The simulation is performed under an accompanying simulation environment on a VAX computer. TAGS simulation can be done for any valid TAGS design, and at alternative specification levels (IORTDs) without having a fully completed design. The user must provide the simulated input either by interactive input during a simulation run, by data file access, or by additional simulation code representing the system environment.

DCDS will have a similar simulation approach on its VAX implementation. AUTO-G developers state that they are working on AUTO-X, a simulation package to accompany AUTO-G. Cadre is working toward a token based simulation aid for Teamwork by which

the designer can check the expected behavior of the system model in a flow and activation sense.

SADMT is an avenue to overcome such tool limitations and to achieve a simulation environment [Cohen] that can be controlled by the Strategic Defense Initiative Organization for objective and uniform assessment of contractors' BM/C^3 designs.

3.7 SCALE OF DESIGNED SYSTEM

Because the SDI BM/C^3 will be a large and complex design, a crucial question is whether any tool "breaks" (e. g. crashes, runs out of space, or enters some excessively degraded performance mode) as the designed system evolves to its full scale (say, many hundreds of objects, data flows, and decomposition levels). The IDA design exercise was not carried forward sufficiently to answer this, so present users will be contacted regarding their practical project experiences in the near future. This aspect, among others, will be covered in a follow-on report by IDA in 1988.

3.8 EXTENSIBILITY AND ADAPTABILITY

The capability to alter or extend tool functions and design semantics is important for SDI, for several reasons. First, the SDI scale and complexity, and the interests and technical knowledge of different reviewers, call for a means to easily produce non-standard reports on a system design. For similar reasons, new design icons and corresponding extensions of tool semantics, may be important to enhance the graphics and depict the many distinctive attributes of a BM/C^3 system. Also, SDIO is fostering a System Description Language [SRS] employing various graphic presentations that are seldom found all together in available tools.

These requirements imply the following capabilities: access to the tool's database, to add new design attributes and to link in user-written code for processing; access to an icon editor or the tool's display generator, to add new graphic forms; triggers or other means to have user-added functions controlled from the standard tool operations, such as diagnostic checking.

No tool provides all these capabilities. Interactive Development Environments emphasizes its "open architecture" for Software Through Pictures, and already provides

substantial information on its underlying database structure. User-written programs can be used readily with Software Through Pictures. In particular, user-written diagnostics can be triggered from the tool's standard diagnostic operation. A wide range of adaptation also is provided through the ToolInfo file that tailors Software Through Pictures to a user's preferences. PICture, a general drawing tool with predefined icons, is available, but this is not the same as the desired icon editor; only limited tailoring of PIC's icons is available.

Cadre Technologies already provides its Access tool for user-written code to access the Teamwork design database. More development toward an open architecture for Teamwork is forthcoming in releases planned for 1988. A Graphic Notes editor, released with the newest Teamwork version, provides for free-form graphic annotations on diagrams.

TAGS and AUTO-G developers also provide access to their tool database for users who require it, through user-written programs.

DCDS is the only tool that explicitly provides for user-authored design attributes within the specification language. Its QUERY facility also supports user-written validation procedures and reports.

3.9 TEAM DESIGN SUPPORT

BM/C^3 is sufficiently complex that multiple designers must work together concurrently, rather than one at a time using just one database. The tools differ significantly in their mechanisms to support multiple designer teams. An ideal approach is not easily identified, because project scheduling and management enter the picture beside the more technical matters of protection, coordination, and integration of design content.

Protection involves shared, but protected access to parts of a system under design by different designers. Designers will need to view or 'read' others' work, to see system parts or decompositions that affect their own assignments. For both design and database integrity, only one designer may update a working design at one time. Read-only and read/write protection of distinct design objects are essential.

Coordination involves the control of design tasks for mutual consistency, so that their results can be integrated afterward into a self-consistent, composite system design.

Integration involves tool mechanisms that aid this bringing together of results accomplished by different people at different times into one complete system description.

All the tools use protection mechanisms provided by a host operating system, if only to authorize individuals to use the tool. TAGS and Software Through Pictures provide additional access control on design files, allowing read-only or read/write permission to be granted to named individuals for separate system parts. (The applicable "parts" depend on the tool's design semantics.) Teamwork provides read-only and read/write locks, but they are not restricted to specified users. Authorization capability is being added to a future release. AUTO-G relies on Unix operating system protection, supplemented by manual actions of a design administrator. Because of Unix limitations relative to AUTO-G requirements, the design administrator must transfer working copies to other designers. Because of its strictly sequential design method and associated database translations, DCDS does not support concurrent design effort. Basic access protection, if any, would be provided by its host operating system.

Another important need is configuration management, such as TAGS alone provides among these tools. As top levels of a system are developed and approved, they may be placed under formal configuration control within TAGS. Further design, to develop lower levels of detail, references the controlled baseline. Any required revision of the baseline, revealed by the further design, is handled by the formal CM method, wherein the impact across the project can be assessed. Other configuration management tools might be used however, working from tool database information extracted by interface programs. This has been done with Teamwork and STP. Whether done formally with tool support, or informally by a design administrator, CM is crucial to coordination of team efforts.

None of the tools has features that fully automate integrating separate designer contributions into a composite system. The basic copy, edit, and diagnostic features must be invoked manually by an administrator or design manager, in steps that assemble the pieces into a whole and resolve inconsistencies that may slip through prior design coordination. TAGS has a MERGE LIBRARY tool in its Analysis Library package which performs the merge of different system components from other geographical locations into one database. Teamwork has a similar tool. Those with read-only and read/write locks (TAGS, STP, Teamwork), and multiple workstation access to a central design database, in principle avoid having separate products from different designers. Provided the locks are

used judiciously (are not held for long periods), integration is done in place on a single database that all designers access, contributing their individual parts. Administrative control of different versions and baselining (CM) is still essential.

None of the tools has project management support, as would be useful for planning and scheduling multiple designer assignments, or providing other management assistance or reports such as estimation and designer time accounting. Teamwork has a limited facility to enter such information as status annotations on design objects, but users may want to provide their own reporting software to extract or summarize selected data.

4.0 CONCLUSIONS

The IDA exercise has confirmed the usability of these tools, excluding DCDS, for graphic design. It also has highlighted substantial differences among them, across their range of capabilities. In the opinion of IDA evaluators, none of them now provides the full set of capabilities that the SDI program needs.

At this time, TAGS, AUTO-G, and DCDS have greater potential and capabilities in areas most pertinent to SDI specification requirements. In particular, these three provide for timing specification, and for fully defining the behavior of processes. Also, AUTO-G now generates the SADMT representation automatically; TAGS is planned to have this capability in 1988. TAGS has stronger capabilities to support team design, such as configuration management and read/write locking of design parts. TAGS also has its own stand-alone simulation facility; DCDS has simulation in its November, 1987 VAX version, not covered in this report.

Major enhancements are underway for most, if not all of these tools, so present comparisons will change in the months ahead. This evaluation has not covered present user experience thoroughly, and important new experience will emerge from use of these tools by present SDIO contractors in the next few months. Also, IDA did not attempt to stress the tools in terms of design size or complexity. Contractor experience will be of interest on this important issue also. This report is therefore an initial assessment, that will be pursued toward a more substantive recommendation to SDIO in 1988.

REFERENCES

Chen, Peter, Principles of Database Design, Prentice Hall, Englewood Cliffs, NJ, 1985, pp. 174-210, Chapter 5.

Chludzinski, John, A Comparison of Process Design/Description Languages, IDA Paper, forthcoming.

Cohen, Howard, et al., SDI Architecture Dataflow Modeling Technique (SADMT) Simulation Framework, IDA Paper P-2036 (Draft), 17 August 1987.

DeMarco, Tom, Structured Analysis and System Specifications, Yourdon Press, New York, 1978.

Gane, C. and Sarson, T., Structured Systems Analysis: Tools and Techniques, Prentice-Hall, 1979.

Hatley, Derek, "A Structured Analysis Method for Large, Real-Time Systems," Technical Report, Lear Siegler Inc., Instrument Division, Grand Rapids, MI, November 1983.

Heystek, Deborah, Proceedings of the Strategic Defense Initiative Organization (SDIO) Tool Fair, IDA Memorandum Report (Draft), March 1987.

Linn, Cathy Jo, et al., Strategic Defense Initiative Ada Process Description Language Version 1, IDA Paper P-1983, March 1987.

Page-Jones, M., The Practical Guide to Structured Systems Design, Yourdon Press, New York, 1980.

SRS Technologies, Inc., Strategic Defense Initiative System Description Language Specification Version 2 Draft for Comment, 9 July 1987.

Ward, P. and Mellor, S., Structured Development for Real-Time Systems, (2 Vols.) Yourdon Press, 1985.

Zave, Pamela, "An Operational Approach to Requirements Specification for Embedded Systems," IEEE Trans. Software Engineering, 250-269, May 1982.

Appendix A. Tool Summaries

CONTENTS

Technology for the Automated Generation of Systems:

TAGS

of

Teledyne Brown Engineering

TAGS consists of a set of modules that together allow the user to define, analyze, and simulate running of a new system. A system under design, though generally electro-mechanical, normally contains an information subsystem as an important part, and this is the prime part that is usually simulated in its operation. Major components of TAGS are:

a. The Input/Output Requirements Language (IORL). This is the graphical system design language which allows the user to draw block diagrams of the system to be specified, then document the modules and link to other parts of the system.

b. The Diagnostic Analyzer (DA).

c. The simulation system, with the simulation compiler.

TAGS was developed by Teledyne Brown Engineering as part of their Technology for the Automated Generation of Systems (TAGS). At the highest level of design, IORL supports the decomposition of a system into Schematic Block Diagrams (SBD) which use "black boxes" and connectors to symbolize the system's components (processes) and communication between components. Data passed between components are specified (typed) in I/O Parameter Tables. The algorithmic logic that corresponds to an individual component is formally specified in an I/O Requirements and Timing Diagram (IORTD).

Using TAGS, a designer may choose to start with a two-component universe: the system and its environment. Thus, with a Schematic Block Diagram, the system designer first gives a black box description of the system (Figure 1). The components of this block diagram may then be decomposed into further SBDs, as shown in Figure 2, where the INFO_SYSTEM block of Figure 1 is decomposed. The match to inputs and outputs at the higher level is shown through dashed lines at the lower level (the TAGS system requires the specifier to do this, though the analyzer will later check for consistency). As an alternative to decomposition, the required functionality may instead be defined using an I/O Requirements and Timing Diagram (IORTD), as shown in Figure 3.

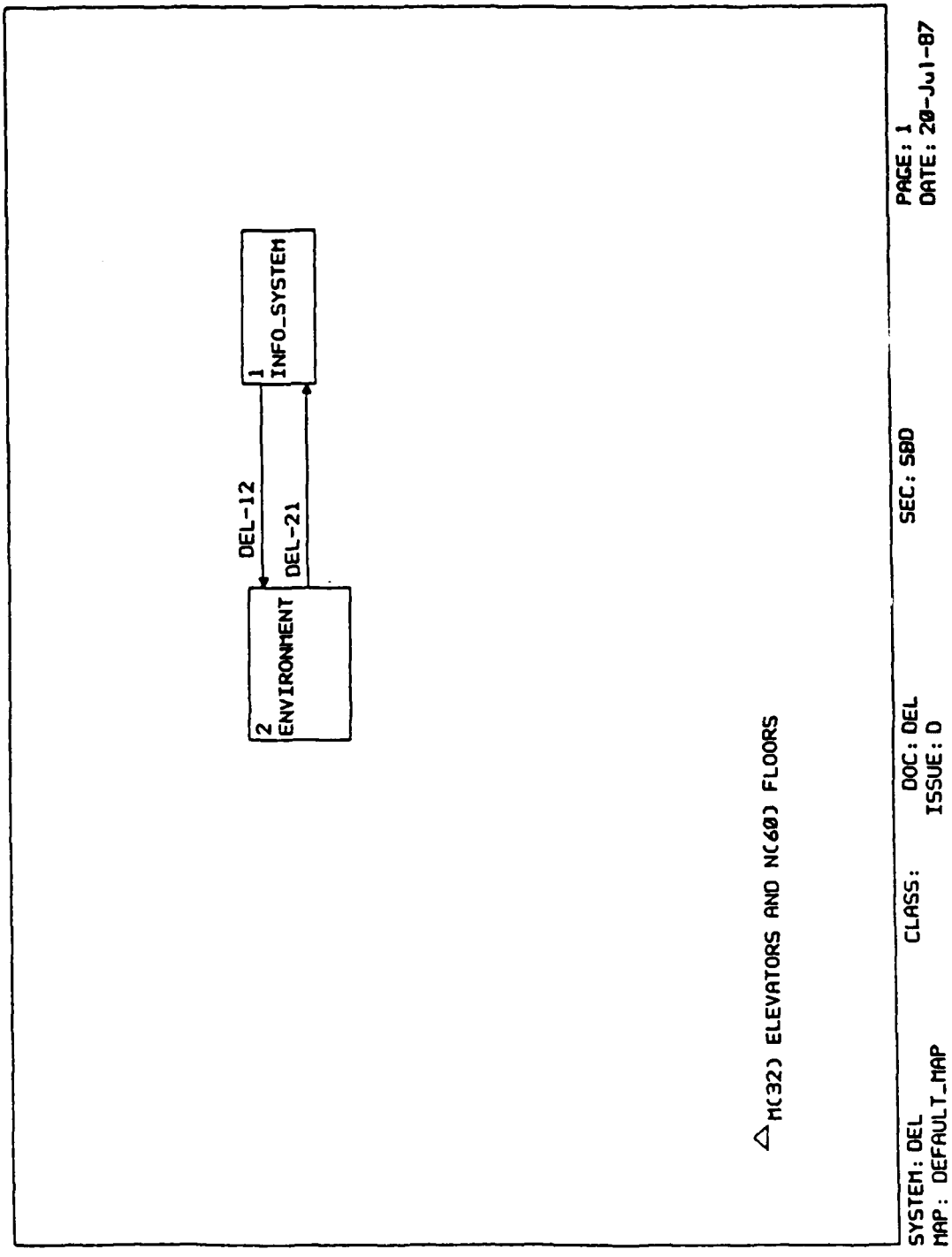


Figure 1 - Schematic Block Diagram (Top Level)

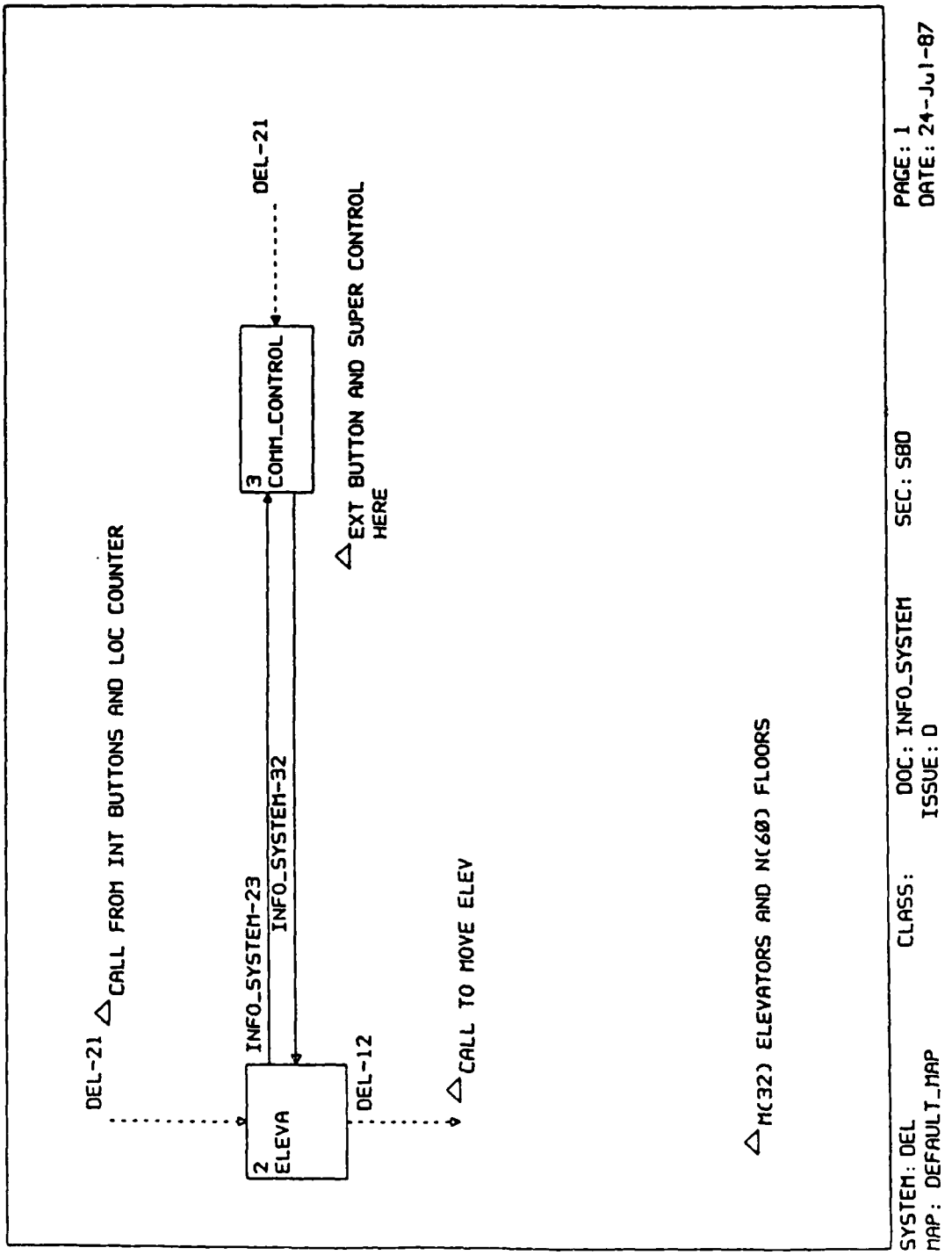


Figure 2 - Schematic Block Diagram (Exploding INFO_SYSTEM)

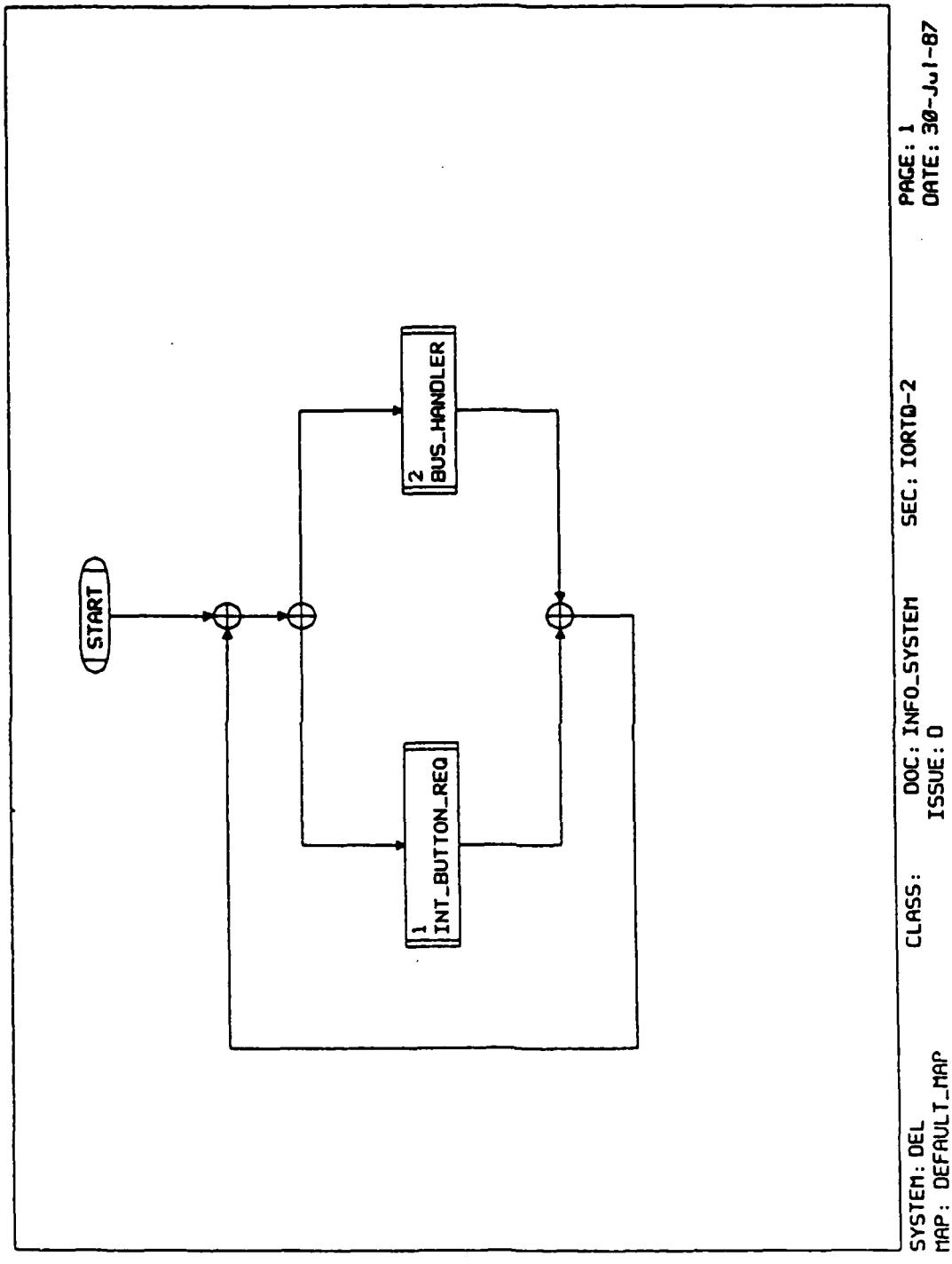


Figure 3 - Input-Output Requirements and Timing Diagram

In this example, the definition of the algorithmic logic for a particular component of the ELEVA box is provided, using two procedure calls: INT_BUTTON_REQUEST and BUS_HANDLER. Communication parameters between components is specified in an I/O Parameter Table (IOPT), as shown in Figure 4, where the line INFO_SYSTEM--32 of Figure 2 is specified. Parameters in these tables may be grouped so that components may transmit groups of parameters.

Components defined by an IORTD are processes. These processes communicate via synchronous "call", "listen", "send", and "receive" events. For a process to send a message it must first call the process with which it is to communicate. The corresponding process must be listening to receive the call. A "call" is a continuous event and begins at a specified time; similarly, the "listen" event is continuous and begins at a specified time, but unlike the "call" event it only lasts for a specified duration. Once a process detects a call from another process it begins sending the specific information. Consequently, the two events must overlap in time for the message to be sent. The macro (MAC-11) of Figure 5 illustrates this; it is used in the Predefined Process Diagram (PPD) of Figure 6, which is the procedure definition corresponding to the BUS_HANDLER used in Figure 3. The IORL includes facilities for the definition of data types and operators; examples of the definition of the data used in the BUS_HANDLER procedure are shown in the Input Parameter Table (IPT) of Figure 7.

TAGS allows the use of traditional mathematical notation, as shown in the subscripting of Figure 6, but it makes no special provisions to specify the underlying execution environment. Teledyne Brown Engineering believes that TAGS is capable of describing hardware systems. However, there is no specification capability to explicitly allocate logical processes (software) to hardware resources.

In IORL, timing requirements are defined in terms of I/O events. Processing and decision logic must occur between the time of the last I/O event and the time of the next I/O event in the control path. Delays may easily be built-in and time tags (internal variables) may be included in the description.

The Diagnostic Analyzer (DA) has three stages, which perform syntactic and semantic analysis, validating the complete specification. Examples from runs of the DA are shown in Figure 8. Provided that the system passed the DA without serious errors, a set of diagnostic and system analysis reports will be automatically generated; these include a variety of hierarchical charts, and are shown on page A-12.

#	PARAMETER DESCRIPTION (U1M)	NAME	VALUE RANGE	UNITS/VALUE MEANING
1				
	SIGNAL	&SIGNAL	(T,F)	
	ELEVATOR POSITION	FLOOR_LOCN	(1,2,...,60)	
		ELEV_NUM	(1,2,...,32)	
3	SUPERVISOR CALLS			
		&EXON	(T,F)	
		START_FL	(1,2,...,59)	
		FINISH_FL	(2,3,...,60)	
		ELEV_NUM		
	SUPER REBOOTS SYS(32)	&REBOOT	(T,F)	
5	EXT FLOOR CALL			
		$UPDOWN	("UP","DOWN")	[2,4]
		EXP	(1,2)	
		FLOOR_CALL	(1,2,...,60)	
6	ELEV INTERNAL CALL			
		WHO	(1,2,...,32)	
		FLOOR_CALL		
7	BUS FAULTS			
		&ONOFF	(T,F)	
		ELEV_DEAD	(1,2,...,32)	

SYSTEM: DEL CLASS: DOC: INFO_SYSTEM SEC: IOPT-32 PAGE: 1
MAP: DEFAULT_MAP ISSUE: 0 DATE: 31-Jul-87

Figure 4 - I/O Parameter Table

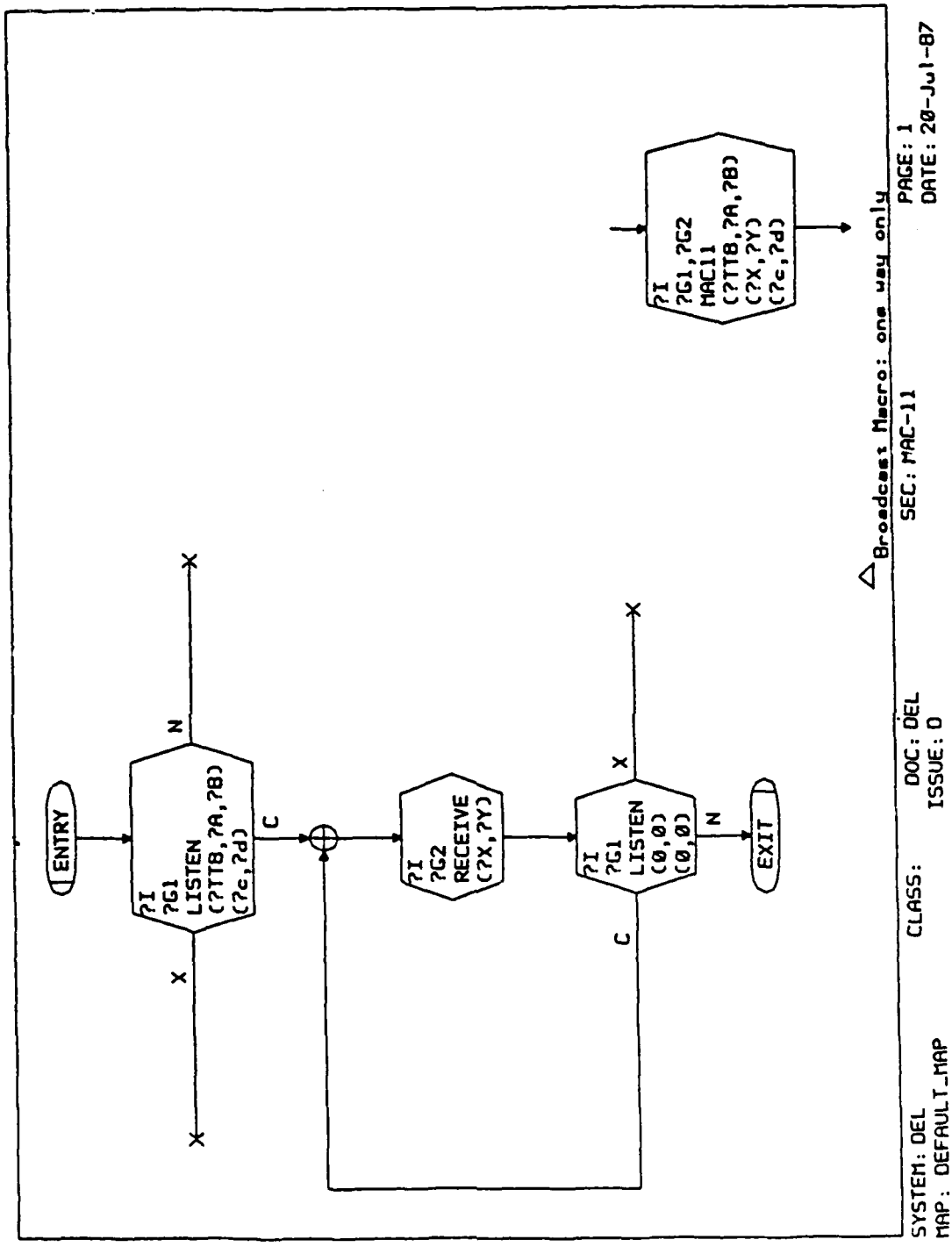


Figure 5 - Macro-11 Defining Communication

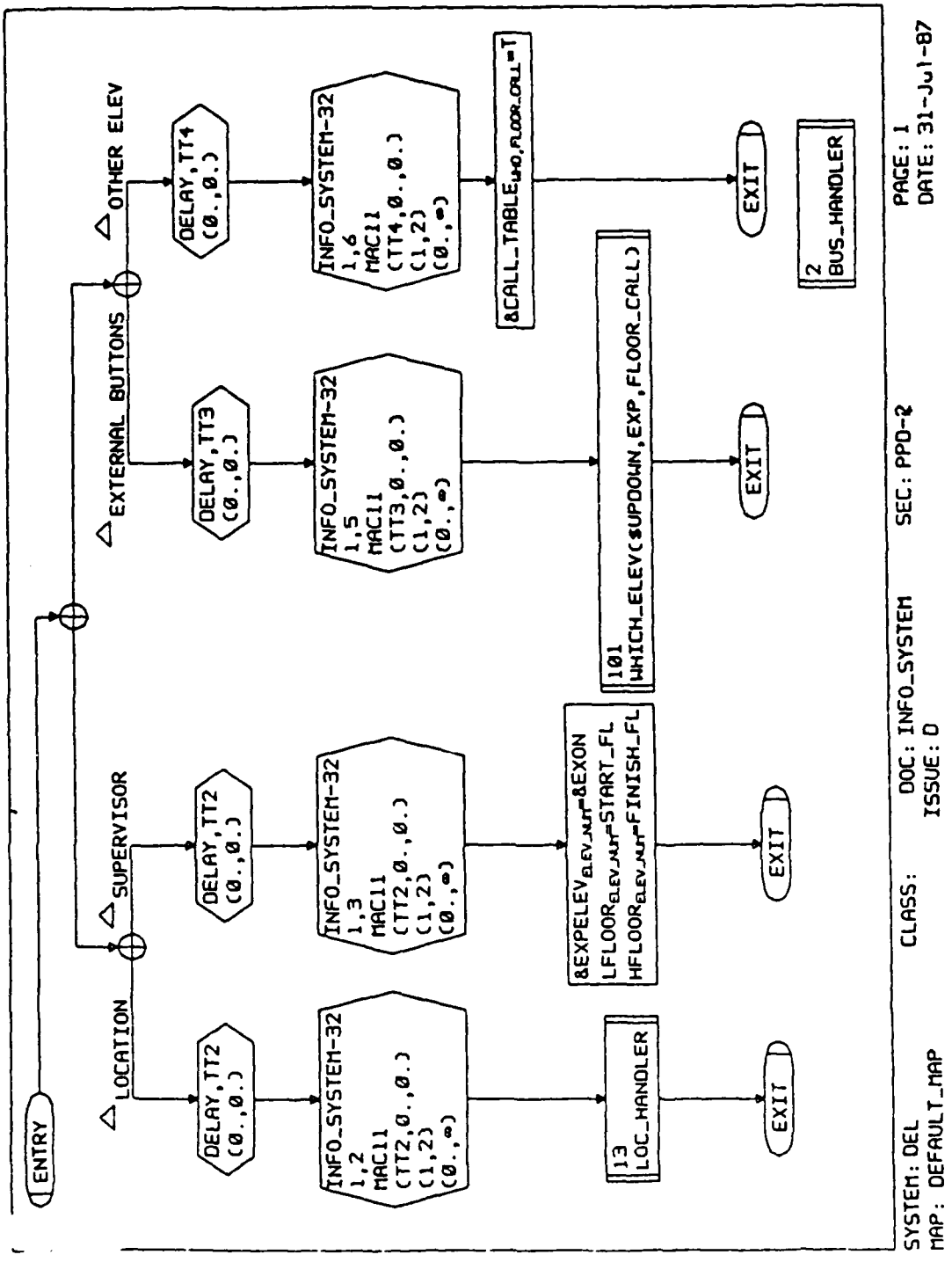
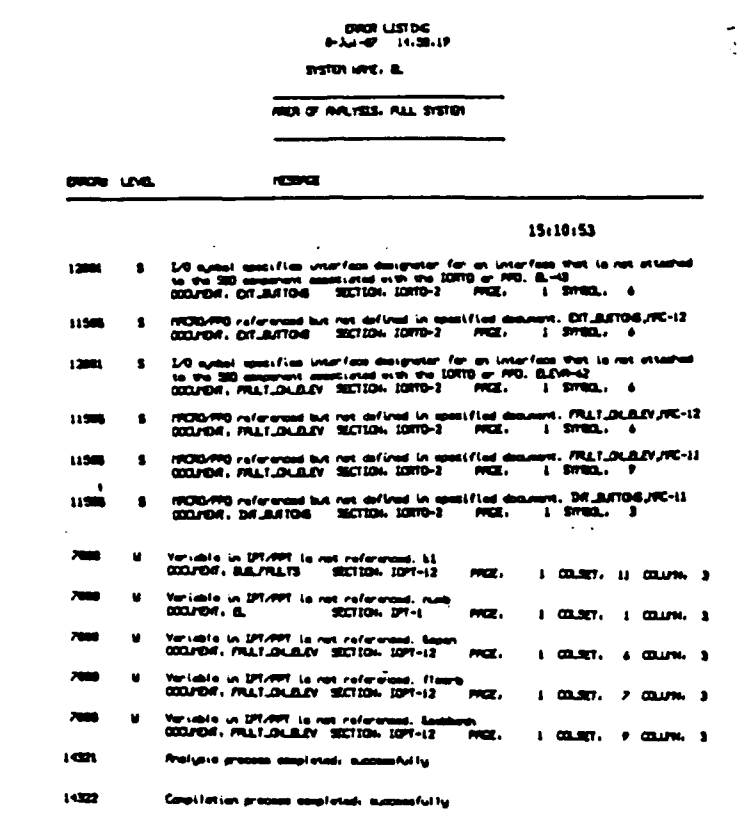


Figure 6 - Predefined Process Diagram Using MAC-11

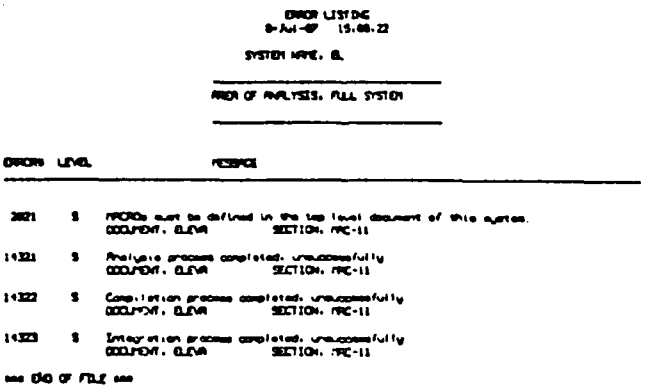
PARAMETER DESCRIPTION (DIM)	NAME	VALUE RANGE	UNITS/VALUE MEANING
EXPRESS ELEVATOR(32)	&EXPELEV	(T,F)	
LO FLOOR(32)	LFLOOR	(1,2,...,59)	
HI FLOOR(32)	HFLOOR	(2,3,...,60)	
INDEX OF ELEV	SELECTED	(1,2,...,32)	
DIRECTION(32)	$DIR	("UP","DOWN", "NEITHER")	[2,7]
POSITION(32)	LOCATION	(1,2,...,60)	
MOTION DELIMIT	&INMOTION	(T,F)	
LOCATIONS CALLED (32,60)	&CALL_TABLE	(T,F)	
SET OF POSS RESPONDERS(32)	&SET	(T,F)	

SYSTEM: DEL CLASS: DOC: INFO_SYSTEM SEC: IPT-2 PAGE: 1
MAP: DEFAULT_MAP ISSUE: D DATE: 31-Jul-87

Figure 7 - Input Parameter Table



```
                    ERROR LISTING
                    6-Jul-87  14:38:19

                    SYSTEM NAME: EL

                    AREA OF ANALYSIS: FULL SYSTEM

ERROR#  LEVEL        MESSAGE
-------------------------------------------------------------------------------

                                                                    15:10:53

12000    S    I/O symbol specifies interface designator for an interface that is not attached
              to the IED component associated with the IORTD or PPD. EL-43
              OCCURRENT: EXT_BUTTONS    SECTION: IORTD-2     PAGE:    1 SYMBOL:   6

11500    S    MICRO/PPD referenced but not defined in specified document. EXT_BUTTONS,MIC-12
              OCCURRENT: EXT_BUTTONS    SECTION: IORTD-2     PAGE:    1 SYMBOL:   6

12001    S    I/O symbol specifies interface designator for an interface that is not attached
              to the IED component associated with the IORTD or PPD. ELEVA-42
              OCCURRENT: FAULT_OR_ELEV  SECTION: IORTD-2     PAGE:    1 SYMBOL:   6

11500    S    MICRO/PPD referenced but not defined in specified document. FAULT_OR_ELEV,MIC-12
              OCCURRENT: FAULT_OR_ELEV  SECTION: IORTD-2     PAGE:    1 SYMBOL:   6

11500    S    MICRO/PPD referenced but not defined in specified document. FAULT_OR_ELEV,MIC-11
              OCCURRENT: FAULT_OR_ELEV  SECTION: IORTD-2     PAGE:    1 SYMBOL:   9

11500    S    MICRO/PPD referenced but not defined in specified document. INT_BUTTONS,MIC-11
              OCCURRENT: INT_BUTTONS    SECTION: IORTD-2     PAGE:    1 SYMBOL:   3

7000     W    Variable in IPT/PPT is not referenced. bl
              OCCURRENT: BUS_FAULTS     SECTION: IOPT-12     PAGE:    1 COLSET:  11 COLUMN:  3

7000     W    Variable in IPT/PPT is not referenced. numb
              OCCURRENT: EL             SECTION: IPT-1       PAGE:    1 COLSET:   1 COLUMN:  3

7000     W    Variable in IPT/PPT is not referenced. Seven
              OCCURRENT: FAULT_OR_ELEV  SECTION: IOPT-12     PAGE:    1 COLSET:   6 COLUMN:  3

7000     W    Variable in IPT/PPT is not referenced. floors
              OCCURRENT: FAULT_OR_ELEV  SECTION: IOPT-12     PAGE:    1 COLSET:   7 COLUMN:  3

7000     W    Variable in IPT/PPT is not referenced. Eastbound
              OCCURRENT: FAULT_OR_ELEV  SECTION: IOPT-12     PAGE:    1 COLSET:   9 COLUMN:  3

14321         Analysis process completed: successfully

14322         Compilation process completed: successfully

14323    S    Integration process completed: unsuccessfully

*** END OF FILE ***
```



```
                    ERROR LISTING
                    6-Jul-87  15:08:22

                    SYSTEM NAME: EL

                    AREA OF ANALYSIS: FULL SYSTEM

ERROR#  LEVEL        MESSAGE
-------------------------------------------------------------------------------

 2021    S    MICROs must be defined in the top level document of this system.
              OCCURRENT: ELEVA          SECTION: MIC-11

14321    S    Analysis process completed: unsuccessfully
              OCCURRENT: ELEVA          SECTION: MIC-11

14322    S    Compilation process completed: unsuccessfully
              OCCURRENT: ELEVA          SECTION: MIC-11

14323    S    Integration process completed: unsuccessfully
              OCCURRENT: ELEVA          SECTION: MIC-11

*** END OF FILE ***
```

Figure 8 - Listing from Diagnostic Analyzer

The Page Audit listing; an inventory of the system, with the date that each part was originally input or last changed (Figure 9).

The SBD lattice; the hierarchy of the SBDs (Figure 10).

The Data Dictionary, with lists the variables alphabetically, where defined, their type, and where used (Figure 11).

The Flow Analysis listing, which gives the relationships of input and outputs for each interface group (Figure 12).

The PPD Cross Reference log, which gives a list of all calls to and from procedures (Figure 13).

The simulation compiler translates IORL system code into Ada. The portion to be simulated (various levels may be selected) is termed a Blueprint (see a listing in Figure 14). The Ada generated is provided only for simulation purposes.

Page Audit for System

	Document	Issue	Section	Page	Class	Date
1	DEL	D	IOPT-12	1		27-Jul-87
2	DEL	D	IOPT-21	1		31-Jul-87
3	DEL	D	IOPT-21	2		31-Jul-87
4	DEL	D	MAC-11	1		20-Jul-87
5	DEL	D	MAC-12	1		20-Jul-87
6	DEL	D	SBD	1		20-Jul-87
7	ENVIRONMENT	D	IOPT-111	1		24-Jul-87
8	ENVIRONMENT	D	IOPT-222	1		24-Jul-87
9	ENVIRONMENT	D	IOPT-36	1		24-Jul-87
10	ENVIRONMENT	D	IOPT-555	1		27-Jul-87
11	ENVIRONMENT	D	IOPT-777	1		31-Jul-87
12	ENVIRONMENT	D	IORTD-6	1		24-Jul-87
13	ENVIRONMENT	D	IORTD-7	1		24-Jul-87
14	ENVIRONMENT	D	IORTD-77	1		27-Jul-87
15	ENVIRONMENT	D	IPT-6	1		24-Jul-87
16	ENVIRONMENT	D	PPD-1	1		24-Jul-87
17	ENVIRONMENT	D	PPD-11	1		28-Jul-87
18	ENVIRONMENT	D	PPT-11	1		24-Jul-87
19	ENVIRONMENT	D	SBD	1		31-Jul-87
20	EXT_BUTT_PUSH	D	IOPT-12	1		24-Jul-87
21	EXT_BUTT_PUSH	D	IORTD-1	1		20-Jul-87
22	EXT_BUTT_PUSH	D	IORTD-2	1		24-Jul-87
23	EXT_BUTT_PUSH	D	IPT-2	1		24-Jul-87
24	EXT_BUTT_PUSH	D	PPD-1	1		24-Jul-87
25	EXT_BUTT_PUSH	D	PPD-2	1		24-Jul-87
26	EXT_BUTT_PUSH	D	PPT-1	1		24-Jul-87
27	EXT_BUTT_PUSH	D	PPT-2	1		24-Jul-87
28	EXT_BUTT_PUSH	D	SBD	1		24-Jul-87
29	INFO_SYSTEM	D	IOPT-23	1		21-Jul-87
30	INFO_SYSTEM	D	IOPT-32	1		31-Jul-87
31	INFO_SYSTEM	D	IORTD-2	1		30-Jul-87
32	INFO_SYSTEM	D	IORTD-3	1		27-Jul-87
33	INFO_SYSTEM	D	IPT-2	1		31-Jul-87
34	INFO_SYSTEM	D	PPD-1	1		31-Jul-87
35	INFO_SYSTEM	D	PPD-101	1		31-Jul-87
36	INFO_SYSTEM	D	PPD-12	1		30-Jul-87
37	INFO_SYSTEM	D	PPD-13	1		30-Jul-87
38	INFO_SYSTEM	D	PPD-2	1		31-Jul-87
39	INFO_SYSTEM	D	PPD-20	1		31-Jul-87
40	INFO_SYSTEM	D	PPD-30	1		22-Jul-87
41	INFO_SYSTEM	D	PPT-101	1		31-Jul-87
42	INFO_SYSTEM	D	PPT-12	1		31-Jul-87
43	INFO_SYSTEM	D	PPT-13	1		31-Jul-87
44	INFO_SYSTEM	D	PPT-20	1		31-Jul-87
45	INFO_SYSTEM	D	PPT-30	1		22-Jul-87
46	INFO_SYSTEM	D	SBD	1		24-Jul-87

Figure 9 - Page Audit for Complete Design

AUDIT LISTING

SYSTEM: DEL
DATE: 31-Jul-87

SBD LATTICE

```
DEL
        ENVIRONMENT
                SUPER_FAULT
                SUPERVISOR
                        HIS_MACHINE
                        SUPERV_GEN
                BUS_ERRORS
                ERR_FILE
                EXT_BUTT_PUSH
                        EXT_BUTT_GEN
                        E_FLR_CALLS
                EXT_BUT_FAULT
                INT_BUT_FAULT
                INT_BUTT_PUSH
                        INT_BUTT_GEN
                        I_FLR_CALLS
                ELEV_FAULT
                ELEV_SIM
        INFO_SYSTEM
                ELEVA
                COMM_CONTROL
```

ORPHAN SBDs

No Orphans Found

Page Audit for System

	Document	Issue	Section	Page	Class	Date
1	DEL	D	IOPT-12	1		27-Jul-87
2	DEL	D	IOPT-21	1		31-Jul-87
3	DEL	D	IOPT-21	2		31-Jul-87
4	DEL	D	MAC-11	1		20-Jul-87
5	DEL	D	MAC-12	1		20-Jul-87
6	DEL	D	SBD	1		20-Jul-87
7	ENVIRONMENT	D	IOPT-111	1		24-Jul-87
8	ENVIRONMENT	D	IOPT-222	1		24-Jul-87
9	ENVIRONMENT	D	IOPT-36	1		24-Jul-87
10	ENVIRONMENT	D	IOPT-555	1		27-Jul-87

Figure 10 - SBD Lattice

DATA DICTIONARY LISTING
Date: 31-Jul-87

DICTIONARY TYPE : ALL VARS/ ALL DEFS
OR INTEGRATION TYPE: FULL SYSTEM
SYSTEM : DEL
MAP : DEFAULT_MAP

VARIABLE NAME	DEFINED	DATA TYPE	REFERENCED BY
`	(INFO_SYSTEM) PPT-30	INTEGER VARIABLE	(INFO_SYSTEM) PPD-30
@dir	(INFO_SYSTEM) IPT-2	STRING VARIABLE	(INFO_SYSTEM) PPD-12 13 20
@stucko	(ENVIRONMENT) IOPT-111	STRING VARIABLE	(INT_BUTT_PUSH) IORTD-2
	(ENVIRONMENT) IOPT-222	STRING VARIABLE	(EXT_BUTT_PUSH) IORTD-2
	(EXT_BUTT_PUSH) IOPT-12	STRING VARIABLE	
@trans	(EXT_BUTT_PUSH) IPT-2	STRING VARIABLE	(EXT_BUTT_PUSH) IORTD-2 PPD-1 2
	(INT_BUTT_PUSH) IPT-2	STRING VARIABLE	(INT_BUTT_PUSH) IORTD-2 PPD-1 2
@updown	(DEL) IOPT-12	STRING VARIABLE	(ENVIRONMENT) PPD-1
			(INFO_SYSTEM) 2 12 13
	(DEL) IOPT-21	STRING VARIABLE	
	(INFO_SYSTEM) IOPT-32	STRING VARIABLE	
@whichway	(INFO_SYSTEM) PPT-101	STRING VARIABLE	(INFO_SYSTEM) PPD-101
@x	(INFO_SYSTEM) PPT-20	STRING VARIABLE	(INFO_SYSTEM) PPD-20
&call_table	(INFO_SYSTEM) IPT-2	LOGICAL VARIABLE	(INFO_SYSTEM) PPD-1 2 12 13 30
&el_trouble	(ENVIRONMENT) IPT-6	LOGICAL VARIABLE	(ENVIRONMENT) IORTD-6 PPD-1 11
&exon	(DEL) IOPT-21	LOGICAL VARIABLE	(INFO_SYSTEM) PPD-2
	(INFO_SYSTEM) IOPT-32	LOGICAL VARIABLE	
	(SUPERVISOR) IOPT-12	LOGICAL VARIABLE	
&expelev	(INFO_SYSTEM) IPT-2	LOGICAL VARIABLE	(INFO_SYSTEM) PPD-2 20
&fault	(ENVIRONMENT) IOPT-36	LOGICAL VARIABLE	(ENVIRONMENT) IORTD-6
&inmotion	(INFO_SYSTEM) IPT-2	LOGICAL VARIABLE	(INFO_SYSTEM) PPD-12 20
&on	(ENVIRONMENT) IOPT-555	LOGICAL VARIABLE	(SUPERVISOR) IORTD-2
&onoff	(DEL) IOPT-21	LOGICAL VARIABLE	
	(ENVIRONMENT) IOPT-777	LOGICAL VARIABLE	
	(INFO_SYSTEM) IOPT-32	LOGICAL VARIABLE	
&reboot	(DEL) IOPT-21	LOGICAL VARIABLE	
	(INFO_SYSTEM) IOPT-32	LOGICAL VARIABLE	
	(SUPERVISOR) IOPT-12	LOGICAL VARIABLE	
&set	(INFO_SYSTEM) IPT-2	LOGICAL VARIABLE	(INFO_SYSTEM) PPD-20

Figure 11 - Data Dictionary Listing

FLOW ANALYSIS LISTING
Date: 31-Jul-87

FLOW ANALYSIS TYPE : ALL INTERFACES
DA INTEGRATION TYPE: FULL SYSTEM
SYSTEM : DEL
MAP : DEFAULT_MAP

INTERFACE GROUP	OUTPUT REFERENCES	INPUT REFERENCES
DEL-12 GROUP 1	(INFO_SYSTEM) PPD-12 13	(ENVIRONMENT) IORTD-6
DEL-12 GROUP 2	(INFO_SYSTEM) PPD-12 13	(ENVIRONMENT) IORTD-6
DEL-12 GROUP 3	(INFO_SYSTEM) PPD-12 13	
DEL-21 GROUP 1	(ENVIRONMENT) IORTD-77 PPD-1	(INFO_SYSTEM) IORTD-3 PPD-1
	(EXT_BUTT_PUSH) IORTD-2 PPD-2	
	(INT_BUTT_PUSH) IORTD-2 PPD-2	
	(SUPERVISOR) IORTD-2	
DEL-21 GROUP 2	(ENVIRONMENT) PPD-1	(INFO_SYSTEM) IORTD-3
	(SUPERVISOR) IORTD-2	
DEL-21 GROUP 3		(INFO_SYSTEM) IORTD-3
DEL-21 GROUP 4		
DEL-21 GROUP 5	(EXT_BUTT_PUSH) IORTD-2	(INFO_SYSTEM) IORTD-3
DEL-21 GROUP 6	(EXT_BUTT_PUSH) PPD-2	(INFO_SYSTEM) IORTD-3 PPD-1
	(INT_BUTT_PUSH) IORTD-2 PPD-2	
DEL-21 GROUP 7	(ENVIRONMENT) IORTD-77	
ENVIRONMENT-111 GROUP 1		(INT_BUTT_PUSH) IORTD-2
ENVIRONMENT-111 GROUP 2		(INT_BUTT_PUSH) IORTD-2
ENVIRONMENT-222 GROUP 1		(EXT_BUTT_PUSH) IORTD-2
ENVIRONMENT-222 GROUP 3		(EXT_BUTT_PUSH) IORTD-2
ENVIRONMENT-36 GROUP 1		(ENVIRONMENT) IORTD-6
ENVIRONMENT-36 GROUP 2		(ENVIRONMENT) IORTD-6
ENVIRONMENT-555 GROUP 1		(SUPERVISOR) IORTD-2
ENVIRONMENT-555 GROUP 4		(SUPERVISOR) IORTD-2
ENVIRONMENT-777 GROUP 1		(ENVIRONMENT) IORTD-77
ENVIRONMENT-777 GROUP 6		(ENVIRONMENT) IORTD-77
EXT_BUTT_PUSH-12 GROUP 1		(EXT_BUTT_PUSH) IORTD-2
EXT_BUTT_PUSH-12 GROUP 2		(EXT_BUTT_PUSH) IORTD-2
INFO_SYSTEM-23 GROUP 1		
INFO_SYSTEM-23 GROUP 2		
INFO_SYSTEM-32 GROUP 1	(INFO_SYSTEM) IORTD-3	(INFO_SYSTEM) PPD-2
INFO_SYSTEM-32 GROUP 2	(INFO_SYSTEM) IORTD-3	(INFO_SYSTEM) PPD-2
INFO_SYSTEM-32 GROUP 3	(INFO_SYSTEM) IORTD-3	(INFO_SYSTEM) PPD-2
INFO_SYSTEM-32 GROUP 4		
INFO_SYSTEM-32 GROUP 5	(INFO_SYSTEM) IORTD-3	(INFO_SYSTEM) PPD-2
INFO_SYSTEM-32 GROUP 6	(INFO_SYSTEM) IORTD-3	
INFO_SYSTEM-32 GROUP 7		
INT_BUTT_PUSH-12 GROUP 1		(INT_BUTT_PUSH) IORTD-2
INT_BUTT_PUSH-12 GROUP 2		(INT_BUTT_PUSH) IORTD-2

Figure 12 - Flow Analysis Listing

PPD CROSS REFERENCE LOG FILE

System : DEL

PPD CROSS REFERENCE started SUCCESSFULLY on 31-Jul-87 09:46:04.00

Document Id: DEL

Section	Description	Calls	Called Fr

Document Id: ENVIRONMENT

Section	Description	Calls	Called Fr
IORTD-6		(ENVIRONMENT)1 11	
PPD-1	LOCATION_CALC		(ENVIRONMENT)IORTD-6
PPD-11	INIT		(ENVIRONMENT)IORTD-6

Document Id: EXT_BUTT_PUSH

Section	Description	Calls	Called Fr
IORTD-2		(EXT_BUTT_PUSH)1 2	
PPD-1	INITIALIZE		(EXT_BUTT_PUSH)IORTD-2
PPD-2	DAEMON_STICK		(EXT_BUTT_PUSH)IORTD-2

Document Id: INFO_SYSTEM

Section	Description	Calls	Called Fr
IORTD-2		(INFO_SYSTEM)1 2	
PPD-1	INT_BUTTON_REQ		(INFO_SYSTEM)IORTD-2
PPD-2	BUS_HANDLER	(INFO_SYSTEM)7 13 101	(INFO_SYSTEM)IORTD-2
PPD-7			(INFO_SYSTEM)2
PPD-12	MOVE_ET		(INFO_SYSTEM)101

Figure 13 - PPD Cross Reference Listing

BLUEPRINT LISTING

SYSTEM: DEL BLUEPRINT: QQQ OWNER: OPERATOR DATE/TIME: 31-Jul-87 14:11:52

DOCUMENT	ISSUE	PAGE	CHANGE LEVEL	COMPONENT NAME	NUMBER	TYPE
DEL	D	1		ENVIRONMENT	2	SBD
				INFO_SYSTEM	1	SBD
ENVIRONMENT	D	1		SUPER_FAULT	5	SBD
				SUPERVISOR	55	SBD
				BUS_ERRORS	77	IORTD
				ERR_FILE	7	IORTD
				EXT_BUTT_PUSH	22	SBD
				EXT_BUT_FAULT	2	SBD
				INT_BUT_FAULT	1	SBD
				INT_BUTT_PUSH	11	SBD
				ELEV_FAULT	3	IORTD
				ELEV_SIM	6	IORTD
INFO_SYSTEM	D	1		COMM_CONTROL	3	IORTD
				ELEVA	2	IORTD

Figure 14 - Simulation Blueprint Listing

Auto-G
by
Advanced System Architectures

Auto-G is a computer-aided design system produced by Advanced System Architectures, Ltd. of the United Kingdom. Auto-G supports both G, a graphical language, and T, a textual language, for capturing system designs. G is isomorphic to T and both include all the capabilities of a general purpose programming language. The system was first delivered in May, 1986, and is being used in aerospace, telecommunications, and C^3 applications. For the pupose of evaluating Auto-G, IDA used the system hosted on a Sun-3 workstation.

The top level concepts are "documents" and asynchronous processes. A system's design is decomposed into a set of documents, which in turn may be decomposed into documents. The terminal or leaf documents contain the formal specifications of one or more of the system's asynchronous processes. Figure 1 shows the AUTO-G display window with a top view of the elevator control system, decomposed as a document tree.

In Auto-G, system design is represented as a hierarchy of processes. A process is modelled as an "action network" or state transition diagram. The flow of control through the network defines the process behavior. Nodes in the network represent states of the process and are either waiting or intermediate nodes. A waiting node awaits receipt of a signal which determines the subsequent branch flow of a process. An intermediate node is a pause for evaluating a conditional expression which determines subsequent branch flow. Transitions between nodes define a series of actions. The actions could be a sequence of assignment statements, the sending of a signal, etc. A network corresponding to a process has one entry node or initialization, and no exit nodes, because processes are never-ending. Figure 2 shows the action network for the supervisor control panel (*super_panel*).

Processes communicate by sending and receiving signals, which are packets of typed information. Signals are sent asynchronously and are queued for processing. The interface of each process explicitly defines the set of signal types the process is capable of sending or recieving. A signal is valid only if the signal type is included is the process interface. A process will receive a signal only when it is in a waiting node, and the receipt will start the transition to the next node in the action network. If a signal is delivered to a

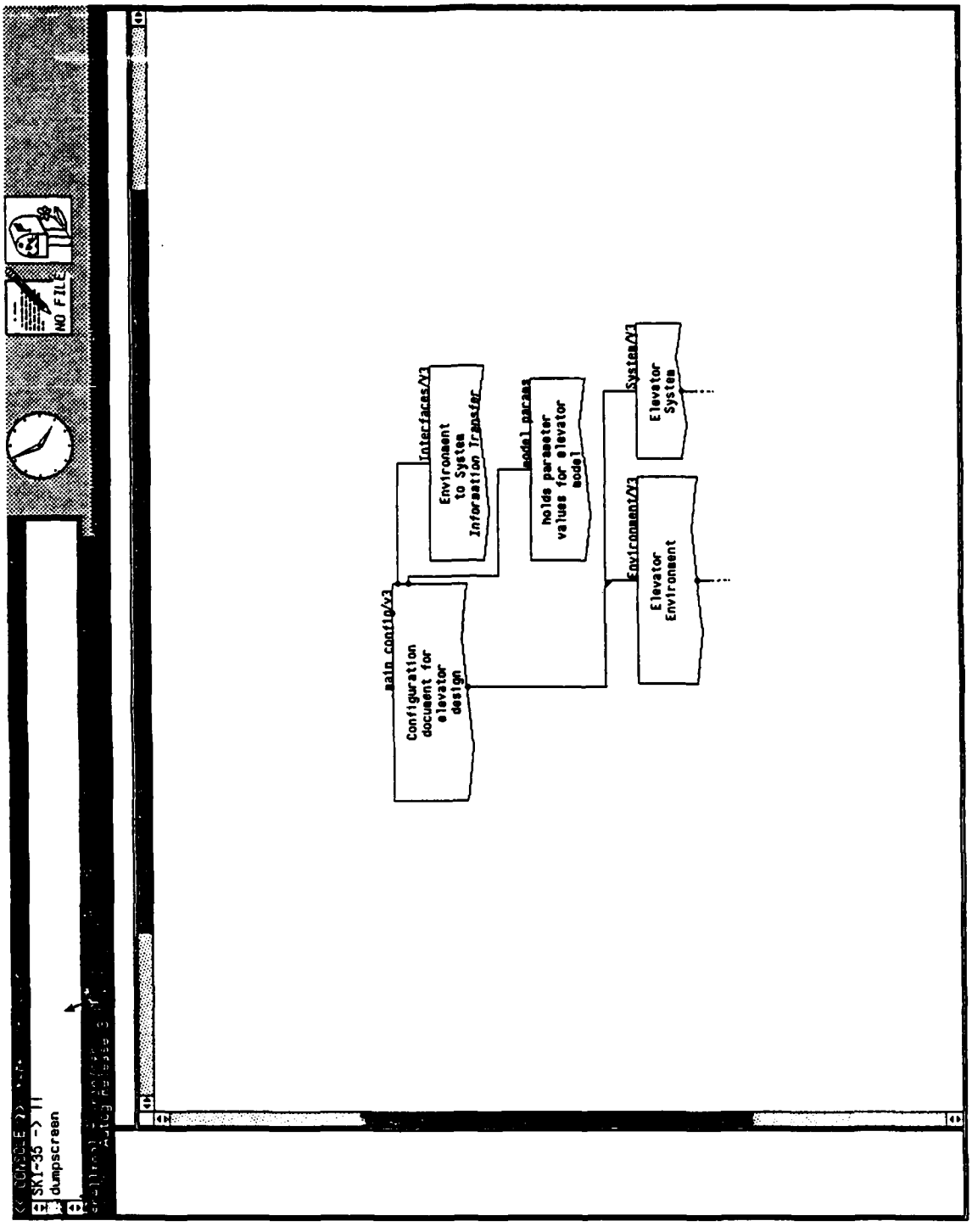


Figure 1 - Editing window with displayed document tree

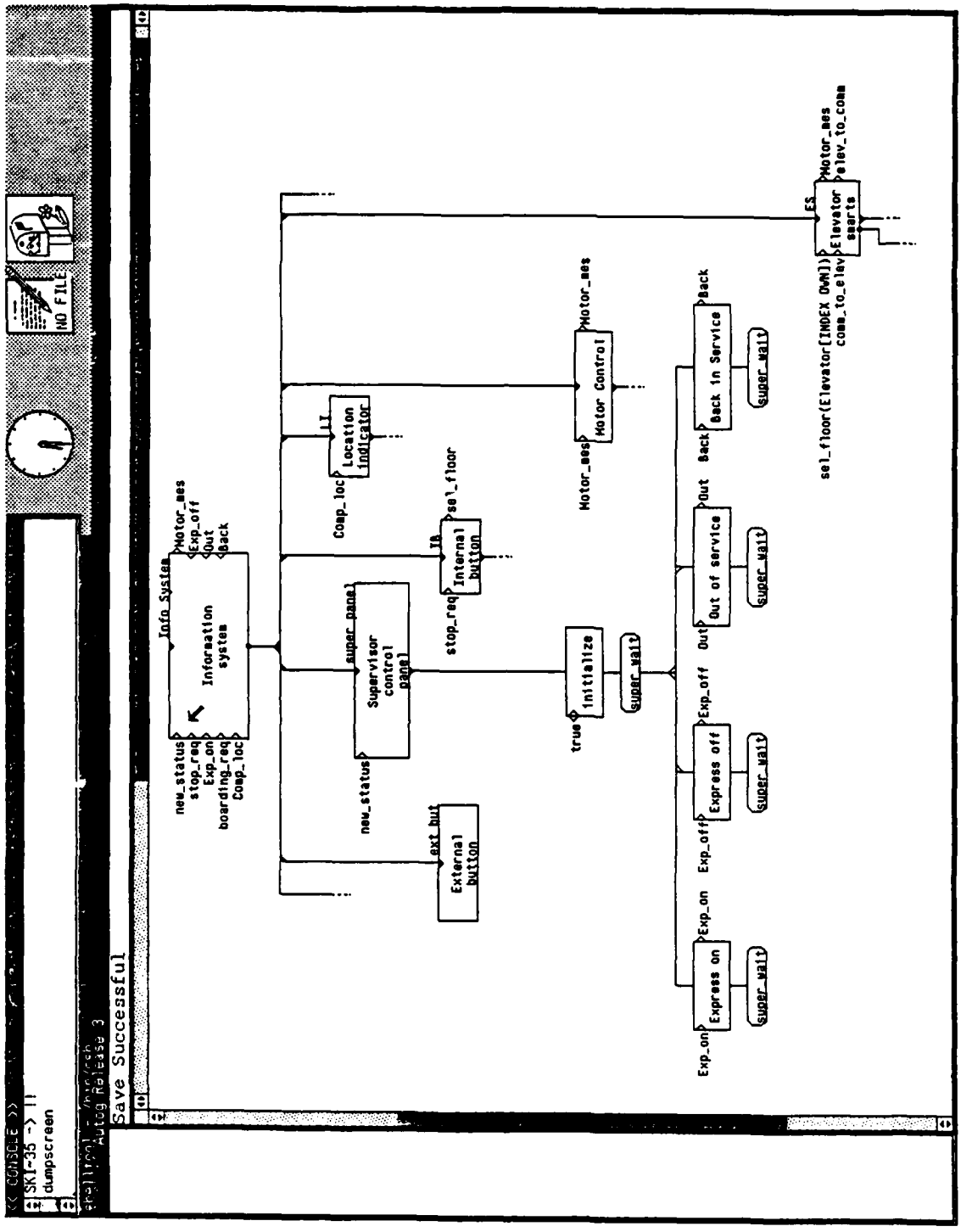


Figure 2 - Process description for onboard controller, with hidden parts

process at a waiting node where the signal is not expected, the signal is discarded. Signals may carry a priority, which affects the relative order of their acceptance by a receiving process.

Using G, the system designer specifies communication with a process's external environment with an "environment list". External objects being read from and/or written to, external signals being sent or received, and external procedures being called are specified in the environment list. The environment list appears on the top level box of the process, as signal identifiers with formal declarations separable into an interface document, as in Figure 3.

Timing and timing constraints are defined in one of two ways. The first way is to specify that a signal sent from a process must not be received until an absolute time (for example, on January 1st, 1988 at 1:00 pm) or a relatively specified time (for example, until 3 seconds has elapsed since sending or receiving a signal).

The second way is to specify that a process waits for one of a number (possibly zero) of specified (types of) signals to arrive and if no such signal arrives within an absolutely or relatively specified time, a specified action is taken.

The first type is typically used to represent periodically repeated functions; whereas the second is typically used to represent constraints. In Figure 4, the action networks for the supervisor control buttons (*elevs_supervisor*) and the passenger request buttons (*passenger_req*) processes specify a random time (*random minutes*) to wait before sending the signals, *new_status* and *stop_req* respectively.

G makes no special provisions for specifying the architecture or hardware of the underlying execution environment for a designed system. G itself could be used as a hardware description language. However, there are no mechanisms for tying the hardware so described to a software system also described in G; i.e., to assign software units or objects to hardware components.

The environment ASA provides for "growing" G-trees (AUTO-G) is adequate (although wearing). Figure 5 provides an example of an icon menu used by the designer for generated trees. Currently, ASA is developing support within the AUTO-G environment for simulation of designs in G or T hosted on the SofChip Processor (ASA's proprietary multi-processor architecture). Further, ASA is finishing work on G-to-SADMT and G-to-C translators; the Rogel Military College is developing a G-to-Ada translator under ASA's auspices (anticipated for early 1988).

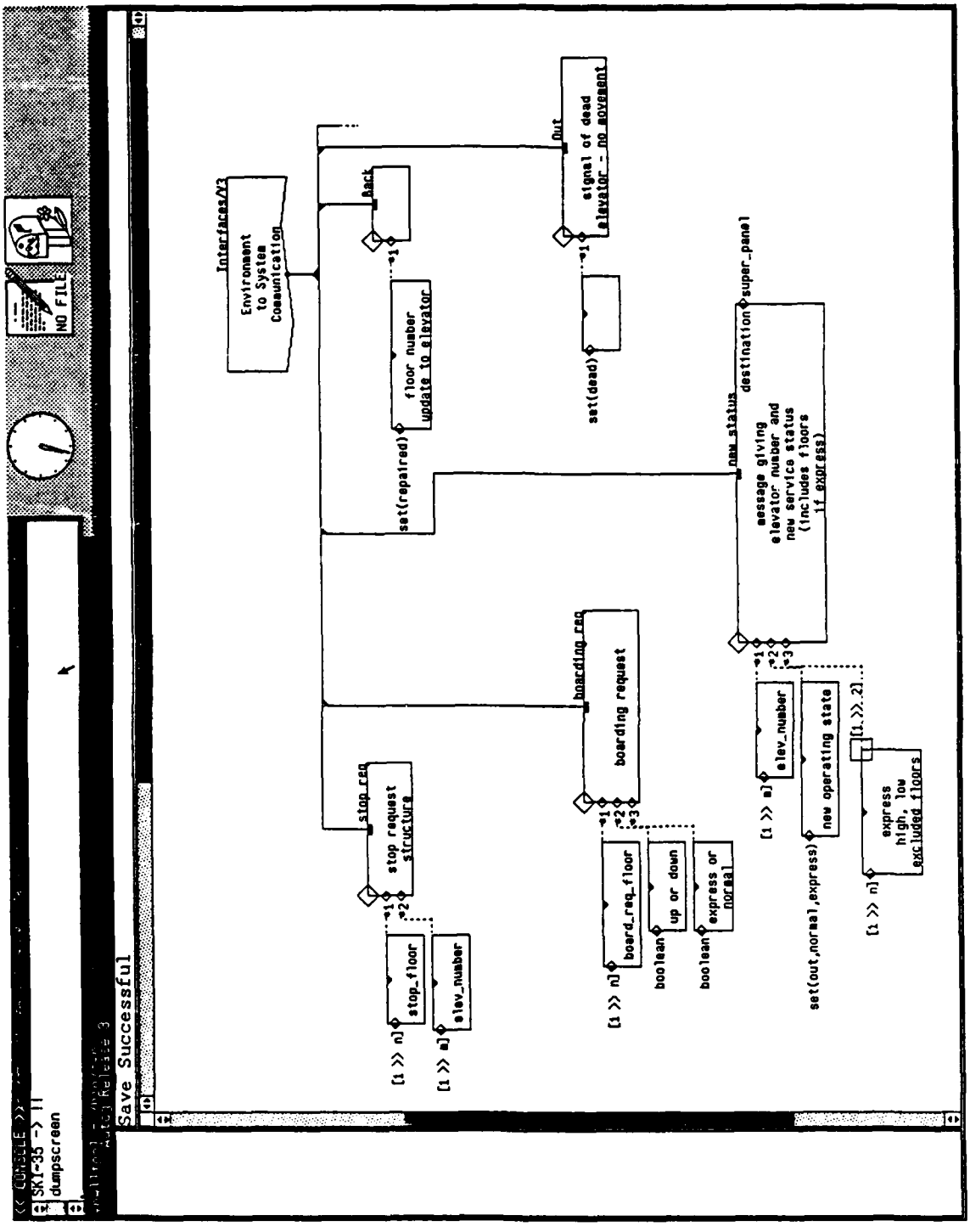


Figure 3 - Process environment list

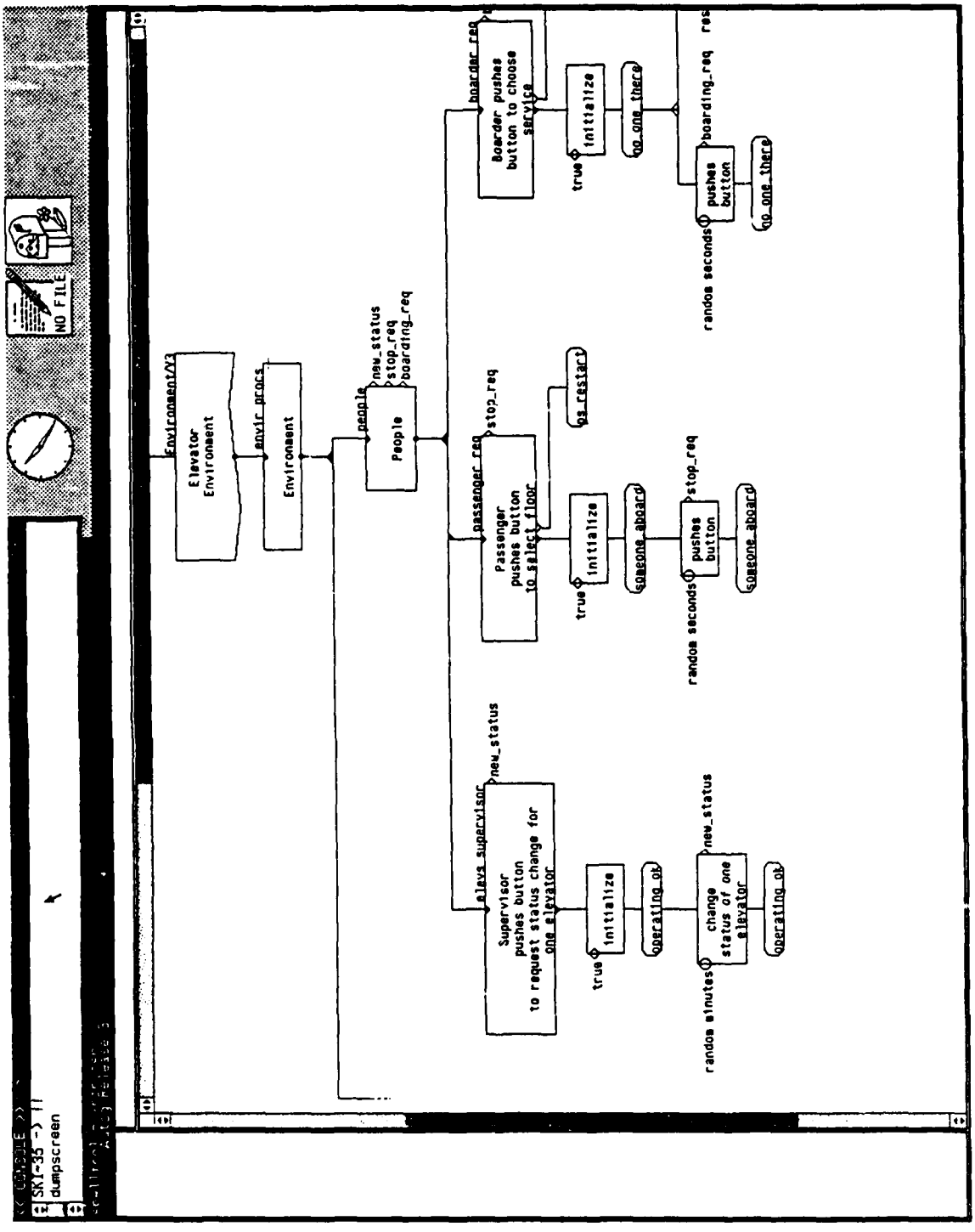


Figure 4 - Interface document content

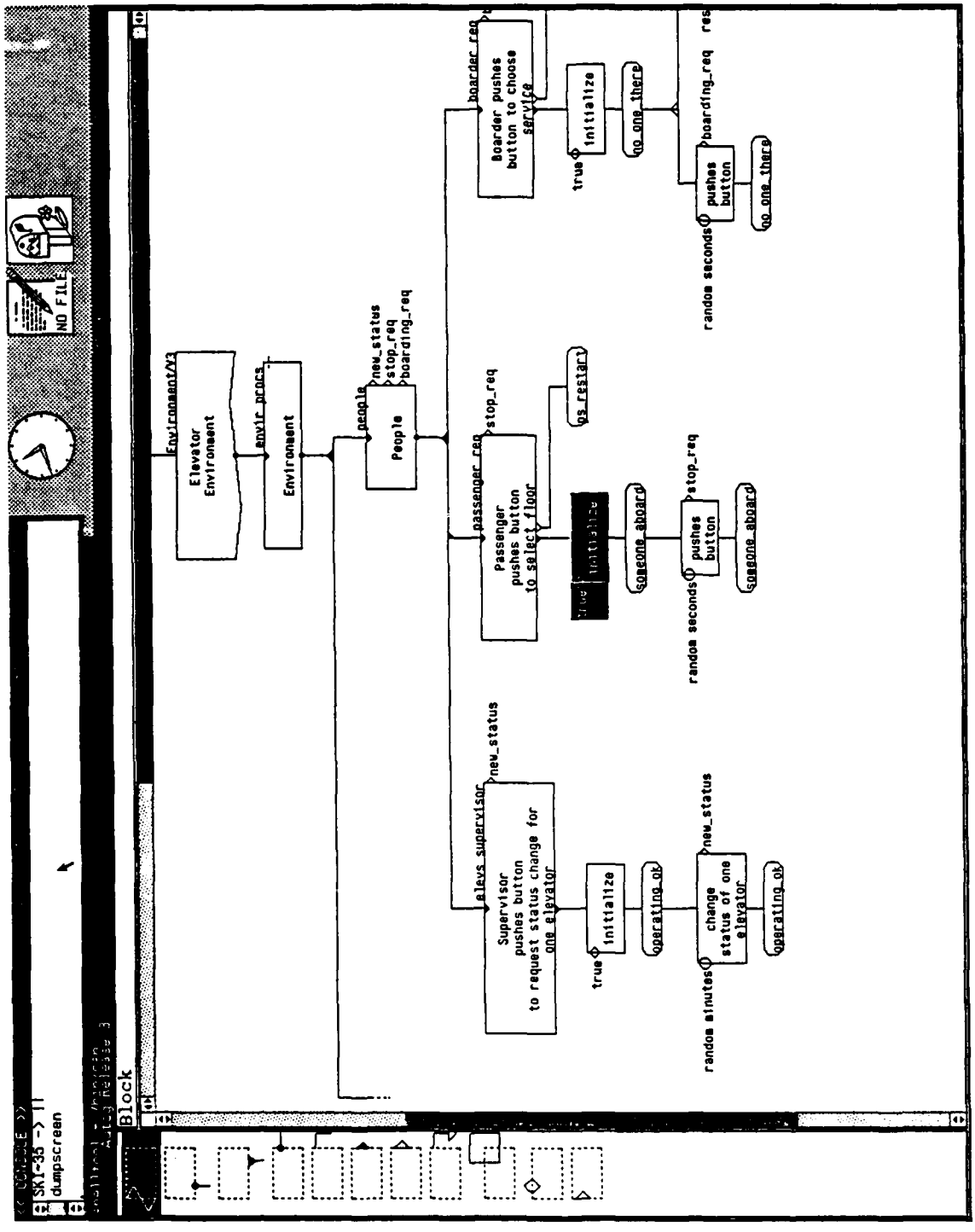


Figure 5 - Timer icons

Distributed Computer Design System: DCDS (TRW and U.S. Army Strategic Defense Command)

DCDS (Distributed Computing Design System) was developed by TRW under contract with the U. S. Army Strategic Defense Command. The primary purpose of DCDS is to develop complete and consistent requirements to provide a foundation for systems development. DCDS has behind it approximately twelve years of development and use on defense projects, dating back to the original SREM system. Two versions of DCDS are being used at this time: a Pascal language version which is being phased out, and an Ada language version which is being beta-tested on the VAX 8600 and the IBM PC-AT. In addition, TRW is developing a simulator to support the first stage of the system development cycle (i.e., System Requirements Engineering Method) based on generated Ada.

DCDS embodies five different design methods and five corresponding design languages:

1) System Requirements Engineering Method	SSL - System Engineering Language, defines systems and their function.
2) Software Requirements Engineering Method	RSL - Software Requirements Engineering Language, defines requirements of S/W.
3) Distributed Design Method	DDL - Distributed Design Language, defines architecture of H/W and S/W.
4) Module Development Method	MDL - Module Development Language, defines units of program code.
5) Test Support Method	TSL - Test Support Language, for testing.

Each of these methods views the system at a different level of detail. Currently, DCDS supports textual versions of all the languages, but System Specification Language (SSL) and Requirements Specification Language (RSL) are the only languages that have graphics editors.

In DCDS, the designer first names the system and its subsystems, and then uses Functional Networks (F_nets) to define a functional model of the system. This decomposition defines a hierarchy of functions and is done using the System Requirements Engineering Method (SYSREM). The designer formally defines the system as the top level function within a F_net. An F_net is used to define the decomposition of a function into sub-functions.

At the next stage of design, using Software Requirements Engineering Method (SREM), the designer works from the Input Interfaces of the primitive functions within the F_net which manipulate data. For each such Input Interface, the designer will define a Requirement Network (R_net). An R_net is a directed graph defining the flow of control between Alphas (small grain functions) and specifies the system response to events and input messages. R_nets also specify the production of output messages. Inputs and outputs defined in the R_net must be consistent with the Input Interface and Output Interface for the corresponding function of the F_net.

Each R_net defines a separate thread of control and hence, a process. R_nets communicate via shared data, that is, variables, files, and so on. An underlying operating system is assumed to manage this process. Using the Distributed Design Method (DDM), the designer will "cluster" Alphas from different R_nets into single processes for optimal hardware assignments.

Input and output data are defined with Item Networks (I_nets). These networks are used to represent the time sequence and arrangement of data (items).

At the SREM level, the designer can use Assertion Nodes within I_nets to specify boolean expressions relating items in the network. The Assertion is evaluated in the context of the related F_net and an exception is raised if the boolean expression is false.

Timing constraints also may be specified at the SREM level by the use of Validation Paths. Within the R_net, the designer may add Validation Point nodes which can be used to specify different Validation Paths within the network. Using Performance Requirements, specific timing constraints can be defined for specific paths within the R_nets. Additionally, DCDS provides the XQT Estimate construct to specify timing constraints for individual blocks of code.

DDM allows the designer to specify the architecture of the host environment for the data processing portion of the system. This method includes the specification of the execution environment from a geographical basis down to a single processor basis. The hardware design is divided into three levels: Level One is concerned with distributed data processing over a connected network of geographically separate nodes; Level Two is concerned with the internal architecture of individual nodes; and Level Three is concerned with the internal design of computer systems which are based on a common architecture (i.e., Level Two designs may include different types of processors integrated by a common network). The designer allocates specific software objects to specific hardware components.

The principal goal of the DDM is the analysis of Alphas within R_nets to arrive at an optimal "clustering" into Tasks. The primary measure of optimality is the assignment of Tasks and the data they manipulate to the same processing nodes. Additionally, it is important to cluster Alphas within the same flow of control (i.e., R_nets) to the Tasks and hence, the same processing nodes. DCDS Processes manage the execution of Tasks.

Module Development Method (MDM) is used for developing algorithms, and for defining a detailed design from the requirements and specifications that were developed at the SSL, RSL, and DDL levels. Units of tested code are produced using the Module Development Language (MDL).

As soon as a design is completed in a particular language, the design is then translated into the next language in the design cycle (ie., SSL to RSL, RSL to DDL, and DDL to MDL).

Currently, TRW has an environment to support system development using DCDS. The tools in this environment include:

- Editors to support the use of a graphical language for designing F_nets, R_nets, and I_nets.
- Editors to enter textual information.
- An editor for changing, adding or deleting an entity, attribute or relations.
- Consistency and completeness checkers.
- A query system to request information about components of the system.

A simulator supporting SYSREM is available as of November 1987. The simulator will produce an Ada programming language representation of a designed system, for simulation.

Software Through Pictures
by
Interactive Development Environments, Inc.

Software Through Pictures (abbreviated STP in this report) was used by IDA on the Sun 3 workstation. Software Through Pictures offers a collection of tools representing the data flow diagramming methodology espoused by Yourdon, DeMarco, Gane, Sarson, Ward, Hatley, and other published authors.

IDEtool is the interactive graphical tool that provides the user with a single interface for accessing and executing all of the tools provided in the STP environment. It utilizes the windowing system of the host machine, and provides a convenient icon, menu and mouse system for direct interaction with the user.

When invoked, IDEtool displays a startup window offering the highest level of user options. Figure 1 shows this window. The available tools include the Data Flow Editor (DFE), Data Structure Editor (DSE), Transition Diagram Editor (TDE), State Transition Editor (STE), Entity Relationship Editor (ERE), Structure Chart Editor (SCE), Control Structure Editor (CSE), and PICture (PCT). Typically, a user selects an editor or command group, a subcommand and the desired options, then selects the "Execute" button to cause that command to be executed. If a graphical editor, such as DFE, is chosen, a separate editor window is created, leaving the IDEtool window available for input of additional commands. In other cases, a separate program is executed, and the output is placed in the teletype area at the bottom of the IDEtool window.

Information is entered into the data dictionary (the text part of the design data base) via the graphical editors or by a Unix text editor of the user's choice. (Diagrams and text are separated in the database.) Each of these can be invoked directly from the startup window, causing a new window in which the chosen editor can be used to create or edit a diagram. As with the startup window, each editor window has several regions. A typical window is shown in Figure 2.

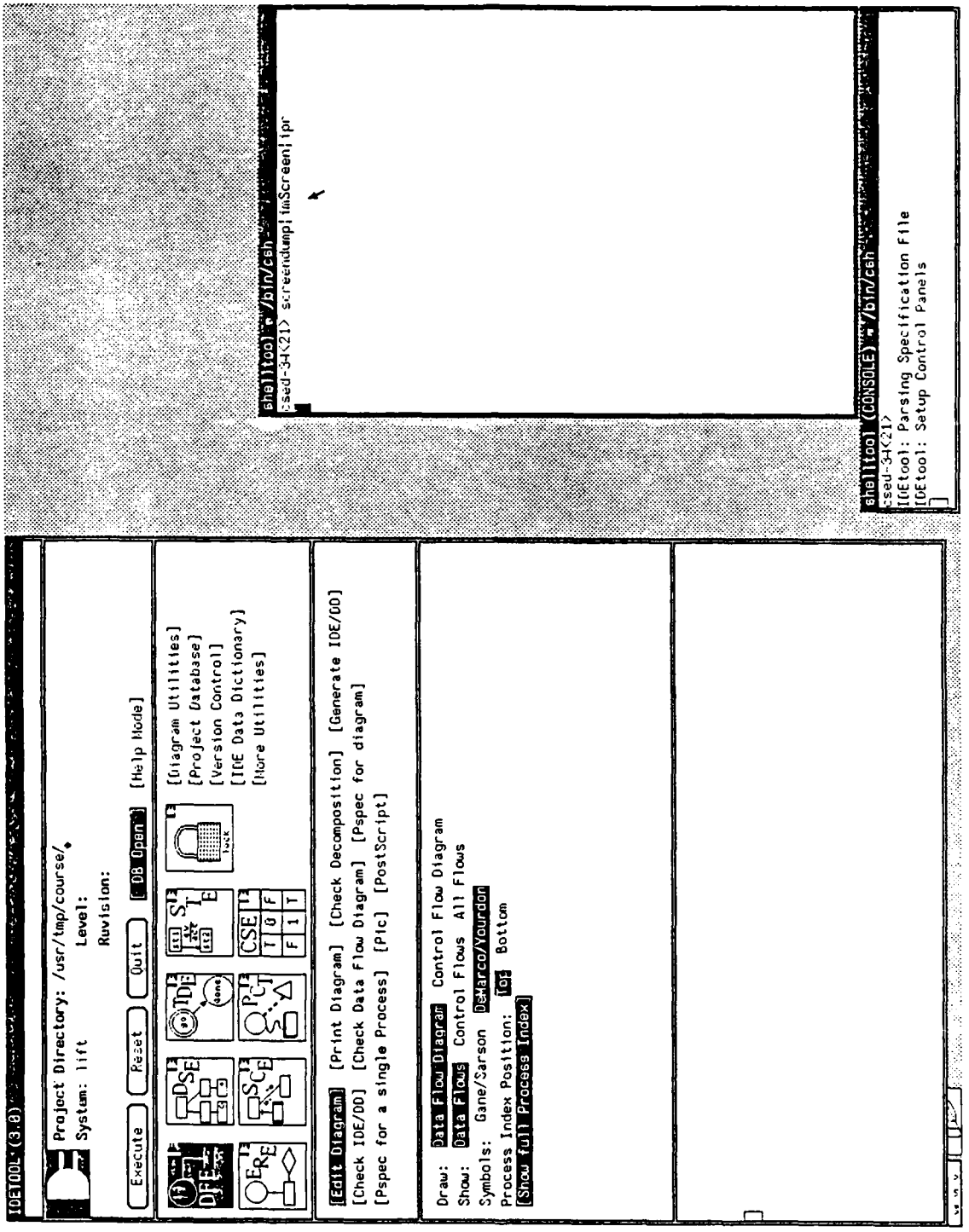


Figure 1 - IDETool Start-Up Window for STP

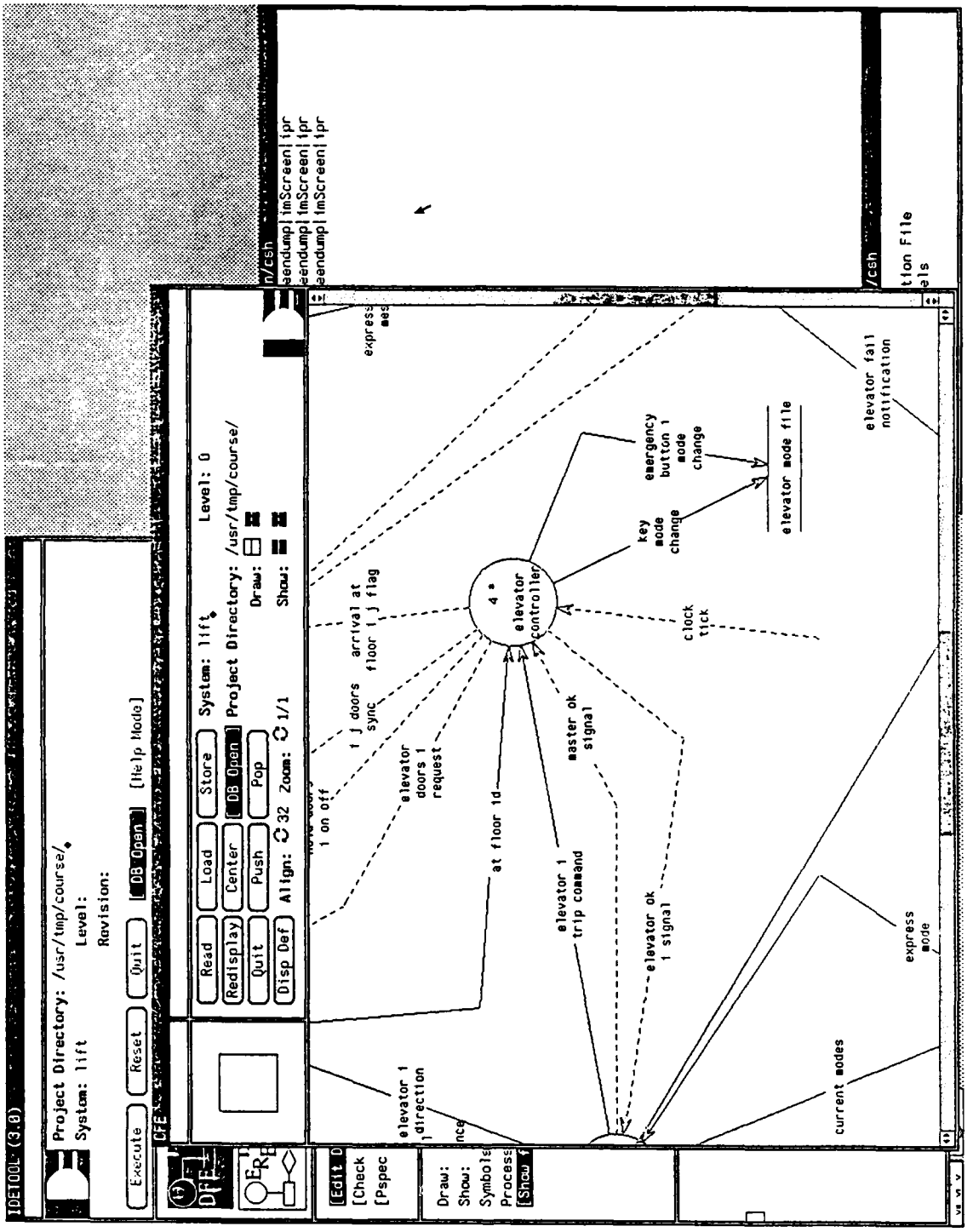


Figure 2 - Composite data flow and control flow diagram in edit mode of the Data Flow Editor

Complexity in design is handled through a hierarchy of diagrams. Graphical editors allow nodes in a diagram to be decomposed into subdiagrams, which are represented as separate diagrams. For example, in the case of Data Flow Diagrams (DFDs), when the user pushes on a process, IDEtool calls up another window for the creation or editing of the DFD decomposing this process. Thus a hierarchy of diagrams can be created. The user moves up and down the hierarchy using the "push" and "pop" buttons in the control panel. Figure 3 shows the next level decomposition, brought up by pushing on the process 4 bubble in Figure 2. Note that control flows are dashed lines and data flows are solid lines. The asterisk inside a process bubble indicates that it is decomposed further. The empty, smaller circles represent source and sink processes on the parent diagram.

Figures 4 and 5 illustrate two of the alternative means available for formally defining control flow. Both apply to the control specification bar at the lower left of Figure 3. Figure 4 is a state transition diagram depicting states of the elevator doors and motor, in response to passenger button pushes and controller commands. Figure 5 depicts the same information in the form of a table.

One of the command groups, IDE Data Dictionary, allows invocation of a menu-oriented program that provides information about items stored in a project database. Although the user can browse the data dictionary using the "Disp Def" button displayed on the control panel of each graphical editor, the data dictionary program is more convenient for extended browsing. Figures 6 and 7 illustrate using the Data Dictionary program, called IDEdd.

The data dictionary program provides the only mechanism currently available for deleting unwanted data dictionaries.

Figures 8 and 9 depict a data structure diagram drawn with DSE, and an error listing as reported by the diagram diagnostic checker.

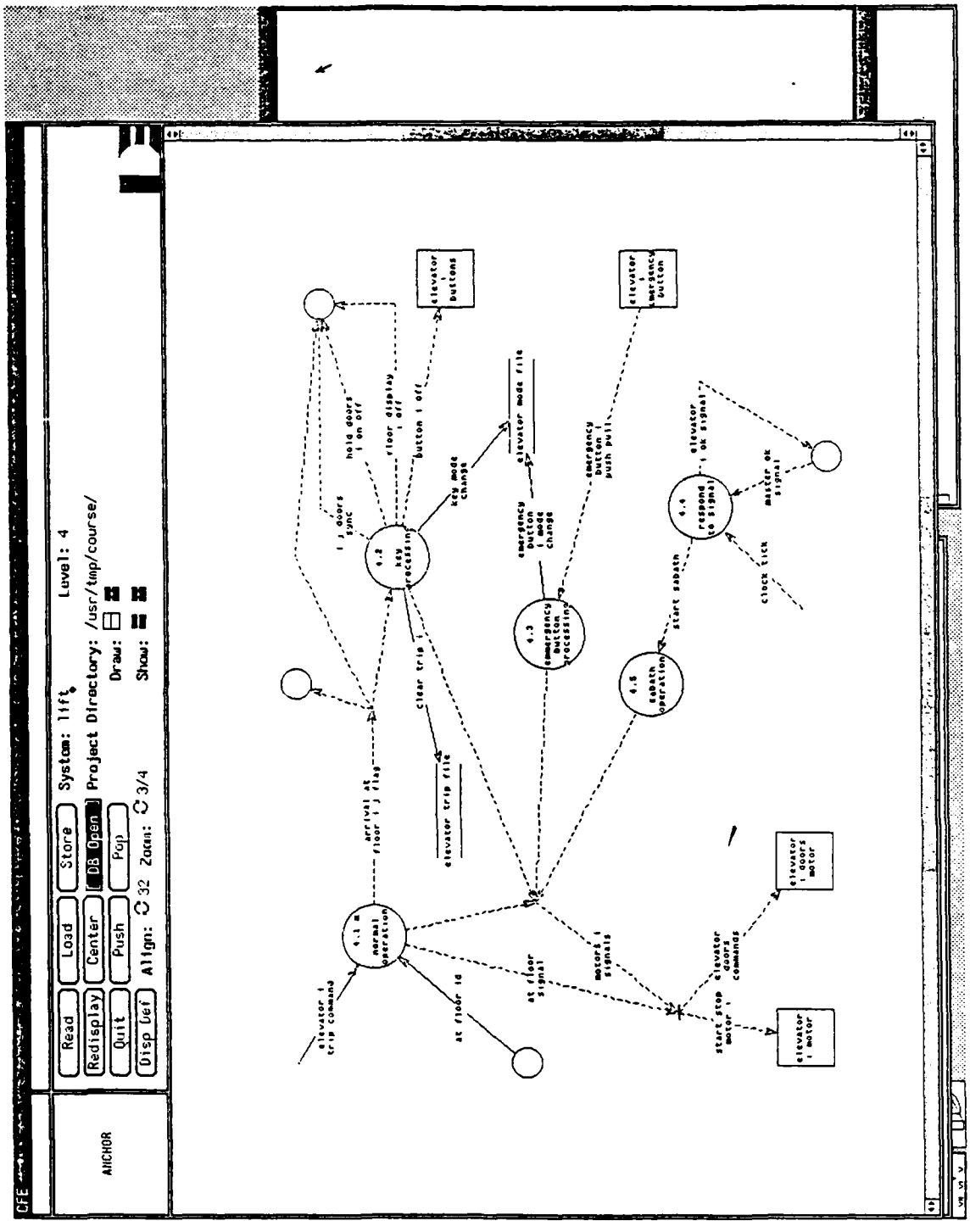


Figure 3 - Next level decomposition of process 4 from Figure 2

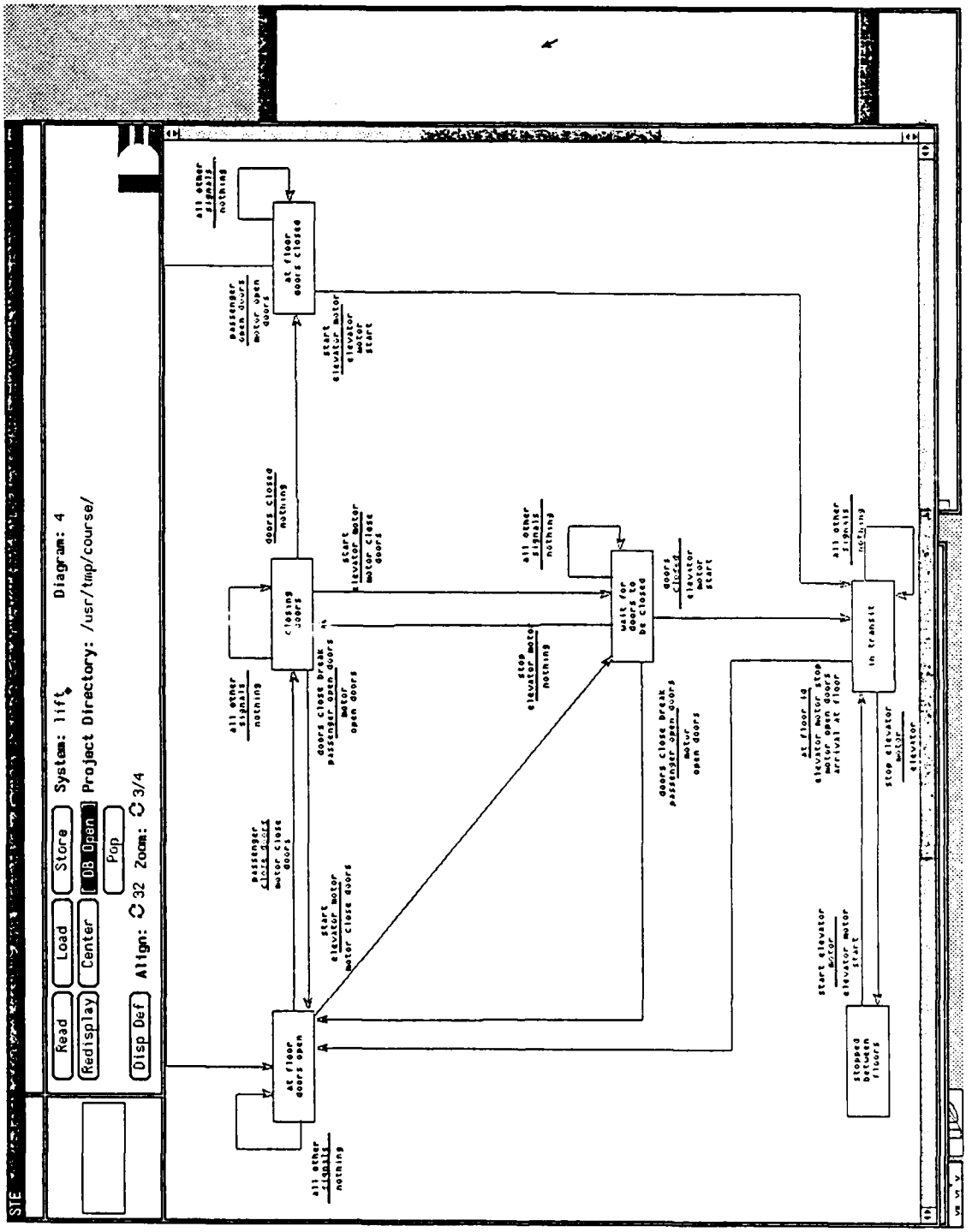


Figure 4 - State transition diagram

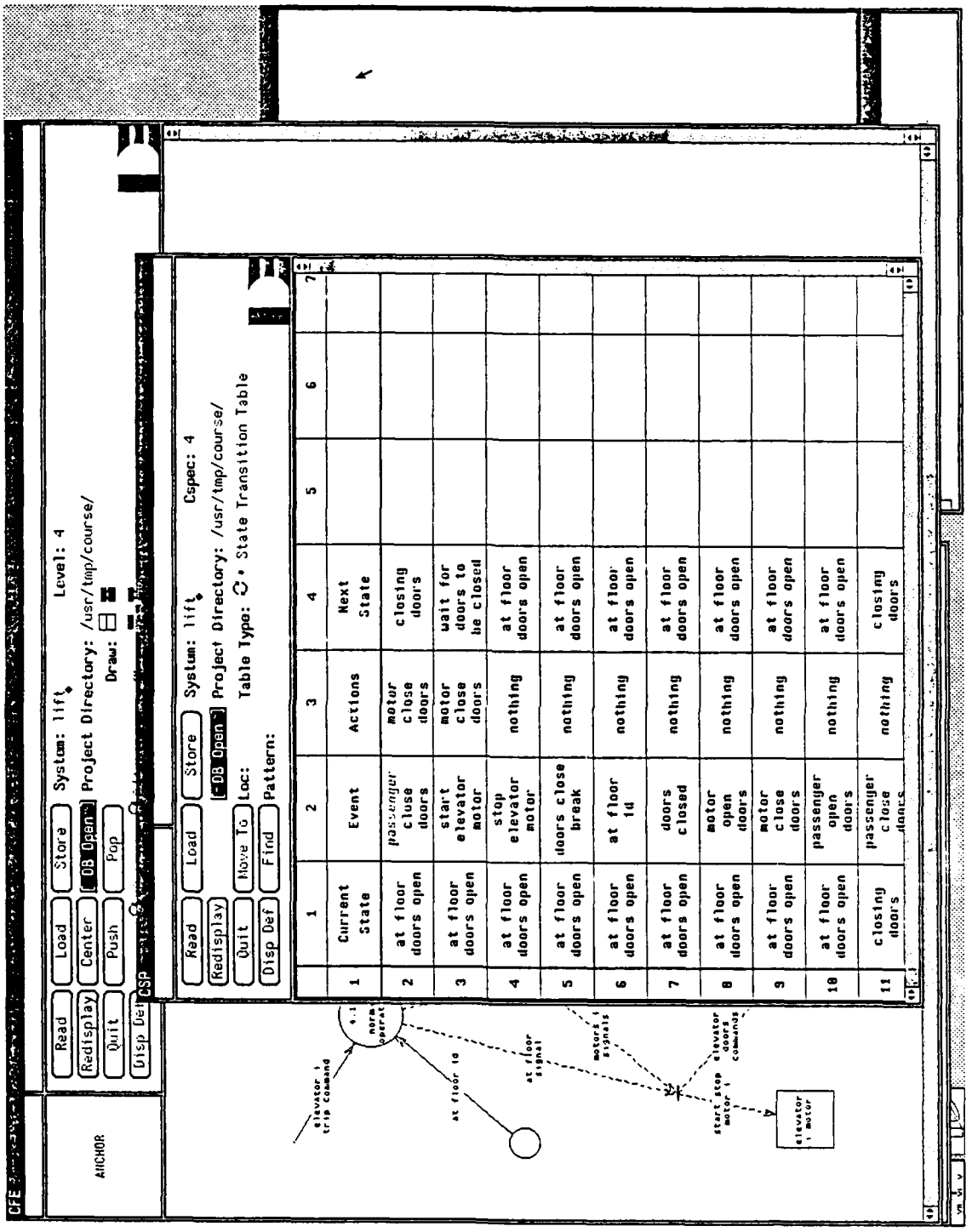


Figure 5 - State transition table

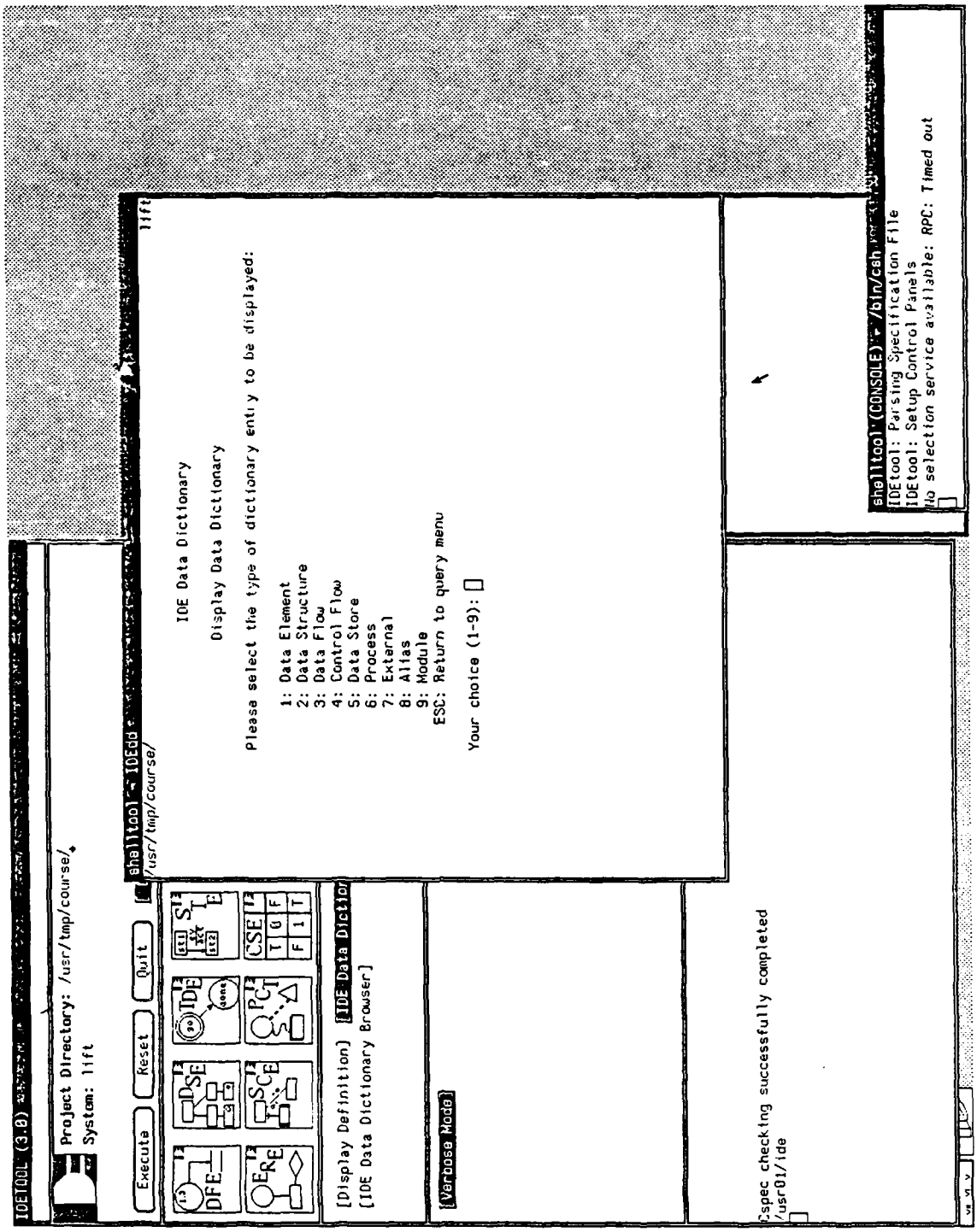


Figure 6 - Query menu of the IDEdd Data Dictionary program

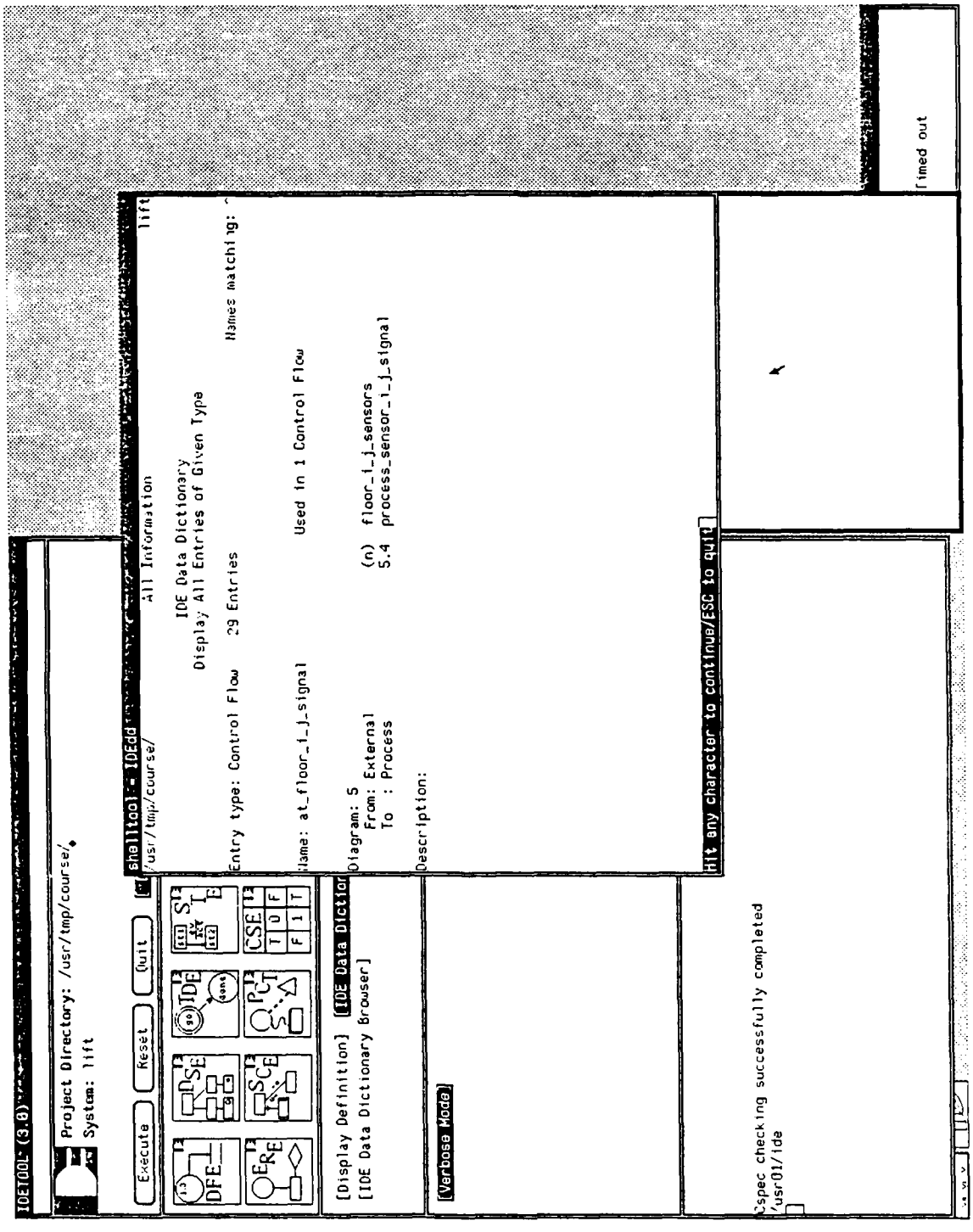


Figure 7 - Query response from IDEdd tool

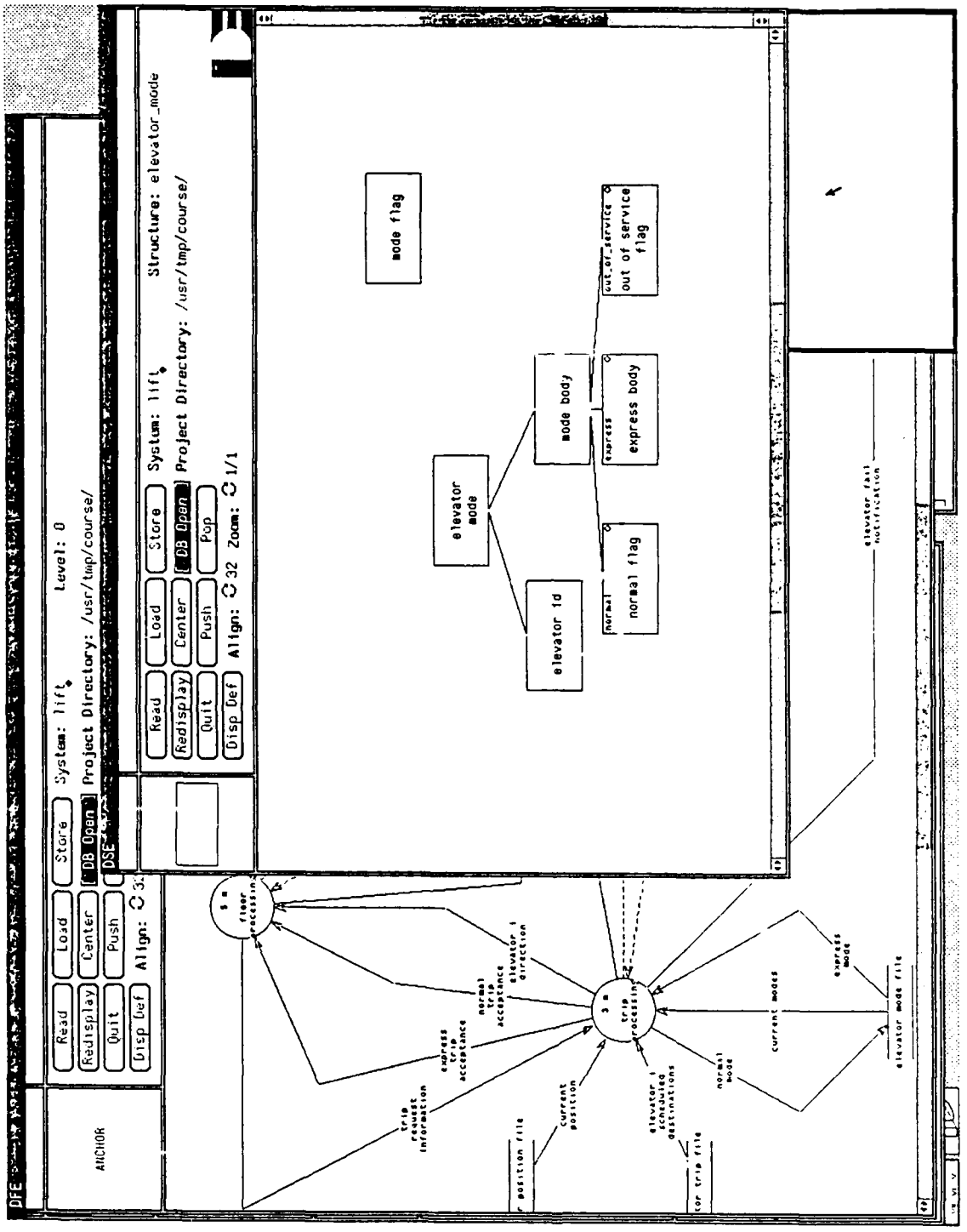


Figure 8 - Data structure diagram

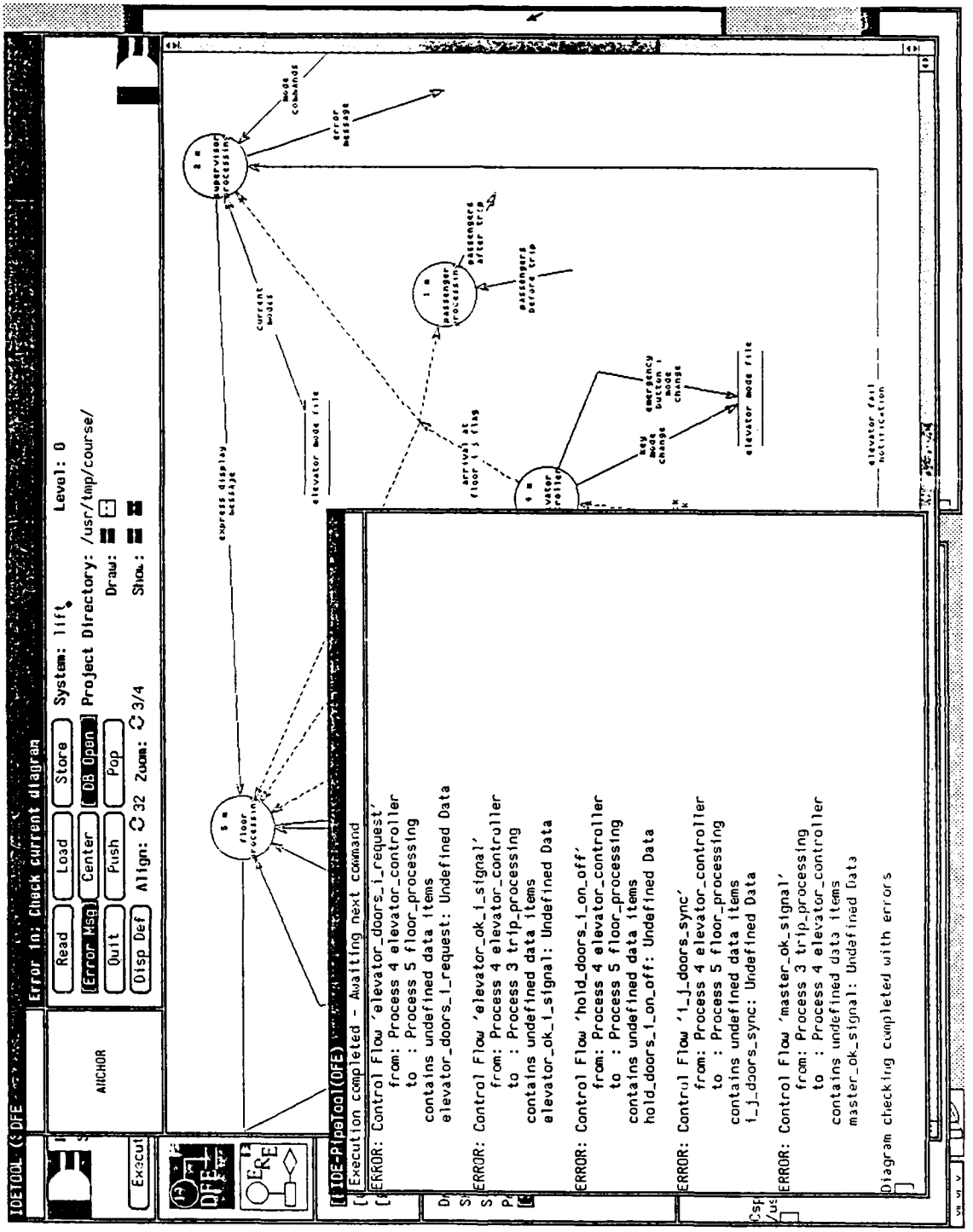


Figure 9 - Error report listing from diagram checking

The PICture editor is somewhat different from the other graphical editors. It provides a larger set of symbols and more flexibility in connecting those symbols. However, since this editor is provided for free-form diagrams, no syntax rules are associated with diagram elements. PICture diagrams cannot be linked to other types of diagrams, nor can diagram information be included in the data dictionary. Figure 10 shows a hardware diagram drawn with PICture.

The STP environment follows an open architecture approach, termed Visible Connections, with published file formats. The open architecture approach has been adopted to allow users to customize the tool by modifying the messages, options, and defaults presented. They can also extend the tool by adding C programs, or by integrating other tools with IDE's.

The ToolInfo file is the central point through which the STP environment may be customized and extended. It is based on the concept of attribute-value pairs, where a value is associated with some attribute of a software tool or its environment. Thus the IDE tool set can be extended by using the defined interfaces to files and databases and/or the underlying tool information file and its supporting library. Although an installation may have a single ToolInfo file which all users use, it is common for a separate ToolInfo to be associated with each project directory, system, or even user. Similarly, multiple specification files can be used.

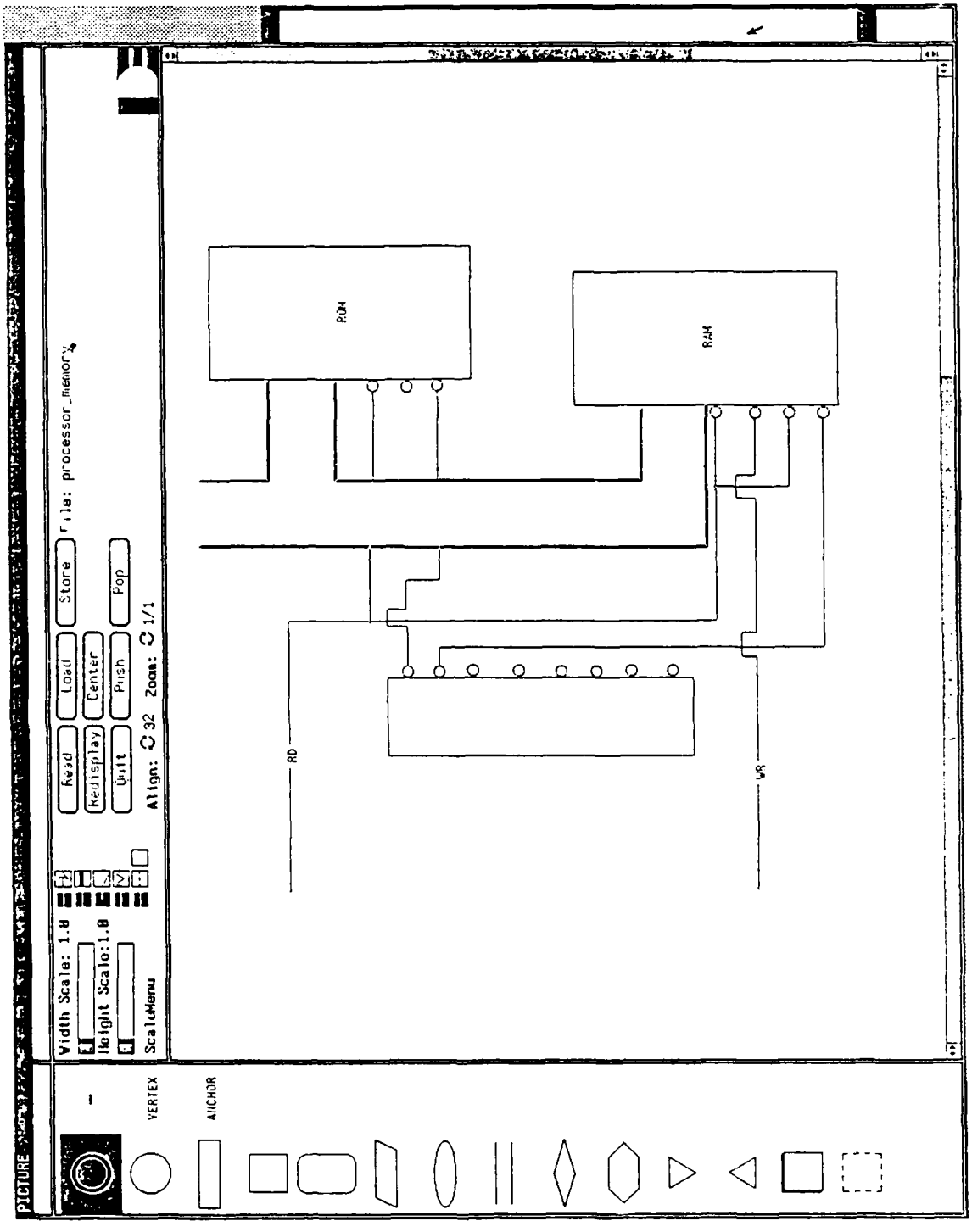


Figure 10 - Hardware diagram drawn with PICture

TEAMWORK
by
Cadre Technologies, Inc.

Teamwork is composed of 6 tool modules:

Teamwork/IM -- Information Modeling (Version 2.2 and above)

Teamwork/SA -- Structured Analysis

Teamwork/RT -- Structured Analysis with Real Time Extensions

Teamwork/SD -- Structured Design

Teamwork/ACCESS -- Utility to Access its Data Dictionary

Teamwork/DPI -- Document Production Interface

Cadre has sold over 1000 copies of Teamwork to several dozen clients. Their largest user is Boeing. Another large user, Federal Express, has implemented over 150 models in the Teamwork package. The largest single model that Cadre is aware of was developed by Lear Siegler. That model consists of over 2400 data dictionary elements and is used by four analysts.

Teamwork, as an integrated product, is a multi-user, multi-window, multi-tasking application that runs on: all Apollo workstations, Sun-2 and Sun-3 workstations, Hewlett Packard 9000-300, DEC VAX station II, and the IBM RT PC. Teamwork can run on the entire VAX family, with its database on any VAX and VAX stations providing the graphics. In addition, Teamwork supports these networks: Ethernet and the Network File System (Sun); networking through the Domain operating system (Apollo); DECnet and Local Area VAX Cluster (LAVC); and various PC and micro-mainframe communication options for the IBM RT PC. Menus, commands, and functionality are identical among versions and independent of the hardware platform. IDA used Teamwork version 2.2 on Apollo.

Teamwork/IM (Information Modeling) provides support for the creation and checking of Chen entity-relationship diagrams for database definition. Although no specific schema generation capability is provided by Teamwork/IM, a user could employ Teamwork's ACCESS package combined with user written C-programs to implement this capability.

Teamwork/SA includes the baseline capabilities necessary to perform system analysis. Formalisms included in Teamwork/SA are data flow diagrams (Yourdon and DeMarco convention) and process specification sheets. Included with Teamwork/SA, as with the other tools, are a centralized data dictionary, a graphical interface, and revision history facility for the design data. In addition, Teamwork/SA provides syntax and consistency checking capabilities for the formalisms it supports.

Teamwork/RT can be purchased in conjunction with Teamwork/SA, and includes capabilities to address the needs of real-time systems designers. These additional capabilities include:

Control Flow Diagrams (as part of DFDs or as separate diagrams)

State Transition Diagrams

Process Activation Tables

Decision Tables

State/Event Matrices

Teamwork/SD is an environment for systems design that works in conjunction with either Teamwork/SA or Teamwork/RT. Formalisms included with Teamwork/SD include structure charts and module specifications. Structure charts included with Teamwork/SD support the methodology prescribed by Constantine and Yourdon and notation as described by Ward and Mellor, and Page-Jones.

Teamwork/ACCESS is a software package that gives the user read (and write in Version 2.3) access to Teamwork's data dictionary through routines written in a higher level language. At this time, the available language binding is C. This open architecture provides the user with the ability to add-on and extend the product. Though significant programming may be required, the Teamwork/ACCESS could be used to:

- Transfer dictionary information between the project dictionary and specialized project management tools (cost & schedule tracking tools, cost estimation tools, etc.)
- Create custom reports that are not provided for by the normal reporting or document production facilities
- Develop custom add-on applications like simulation facilities, code generation facilities, or specialized design metrics.

A feature of Teamwork/ACCESS is its use of read and read/write locking so that custom extensions work in a multi-user environment.

Teamwork/DPI is a package to automatically produce first-cut documents conforming to user-specified templates or to MIL-STD 2167 Data Item Descriptor templates (provided with Teamwork). The package works in conjunction with Teamwork/SD but allows any pictures or text in the Teamwork database, or other ASCII text files on the host platform, to be automatically pulled into the template format. The resulting first-cut documents can then be refined using the Interleaf, Context, or Scribe publishing software available from other vendors. These packages include editing and page formatting capabilities.

The individual tools are integrated under a common graphics user interface. This interface supports multitasking and multiple windows, if this capability is resident in the underlying operating system.

Figure 1 illustrates the initial menu selections available to a user. Figure 1 illustrates the format of the main Teamwork menu bar (Windows...Stop) and the drop down menus which appear under each of the four selections on the main menu. In addition, the "Direct Access..." selection of the second level menu is exploded to a further level as the resulting form is probably the most used selection of the Teamwork menu interface. The "Direct Access" menu provides the mechanism by which a user can:

- Obtain a list of projects (Model Index), a list of diagrams (Process Index), or a list of data dictionary elements
- Open a specific diagram for editing (Object Type:...)
- Annotate any element of the system (project, diagram, or element)

Figures 2 through 9 illustrate some of the diagrams and results which may be accessed through the main menu. These diagrams are as follows.

Syntax and consistency checking is available for each of the above formalisms. Figure 10 illustrates the results of such checking for one Teamwork data flow diagram.

Sample outputs of the data dictionary listing and a typical data dictionary element's attributes are illustrated in Figures 11 and 12. (Figures 1 - 12 follow the text of this evaluation.)

Teamwork has many strengths. It is easy to use and provides a consistent user interface across a variety of hardware platforms and network capabilities, as listed earlier.

Its open architecture allows the user flexibility in extending the data dictionary or linking with adjunct programs for analyses specific to a given design situation. In addition, the document production utility, supporting the automatic merging of text and graphics, facilitates the development of manuals in accordance with DoD specifications. Other features, such as the ability to do an unlimited number of undos or redos during graphical editing, are also very convenient.

Teamwork's newest release, Version 2.3, includes these additional features: a Graphic Notes editor that provides object annotations in free-form graphics (templates also provided for common presentation and flow-chart forms); enhancements to Teamwork/DPI, Teamwork's data dictionary, model renaming and copy features; and, capability to run diagnostic checks as a background process outside Teamwork, as well as to diagnose portions of a model that another user has open.

Teamwork Version 3.0, scheduled for second quarter 1988 release, will include: Model Configuration Management, for establishing baseline designs under formal control with the means to compose a baseline from various component versions; and Teamwork/Ada, a Structure Graph Editor based on Buhr's graphic notations for Ada capabilities. The latter is the first step in an announced plan for Ada support, which will lead to other Ada tools including an Ada Code Generator.

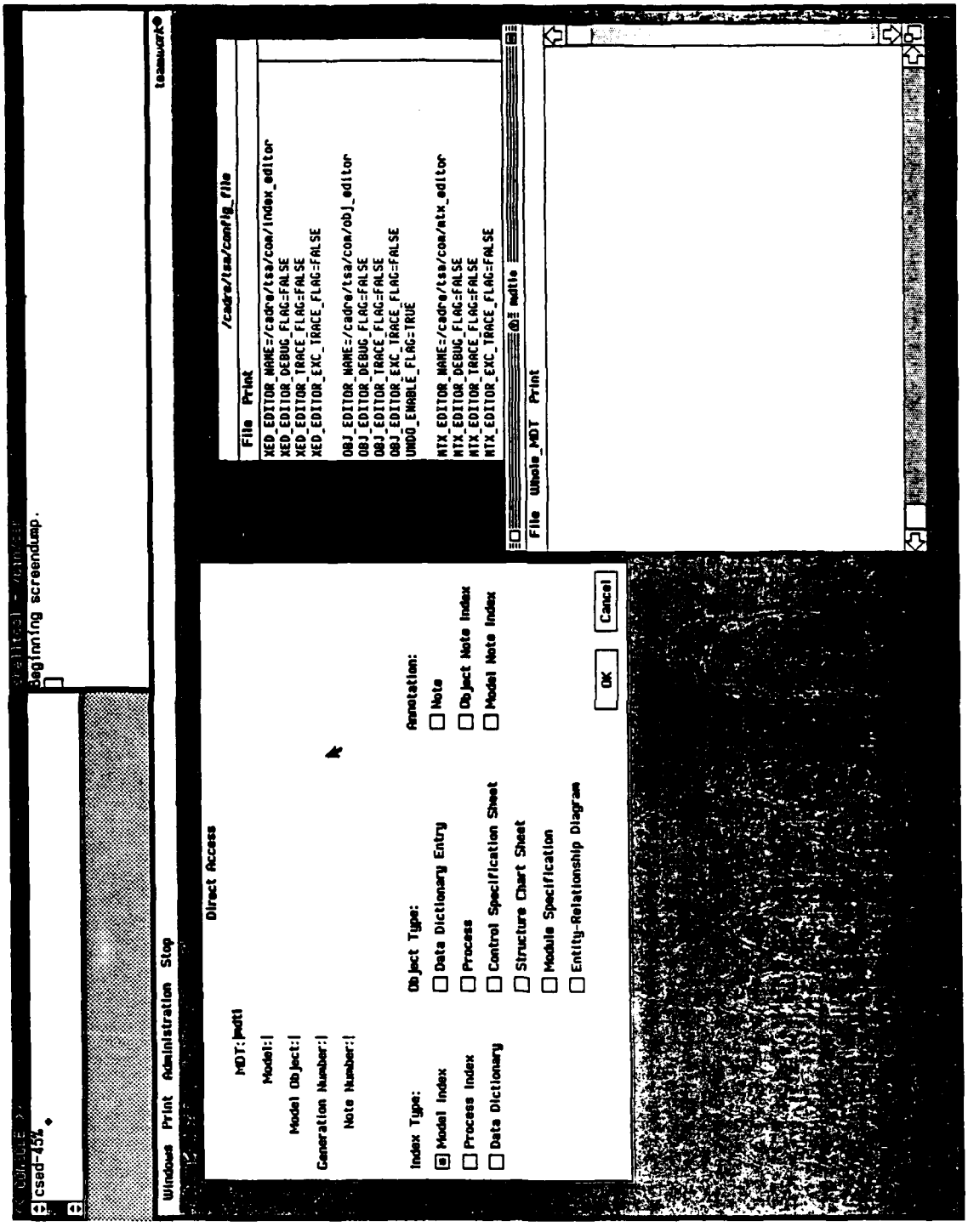


Figure 1 - Main Teamwork Window and Menu, with several subordinate windows opened.

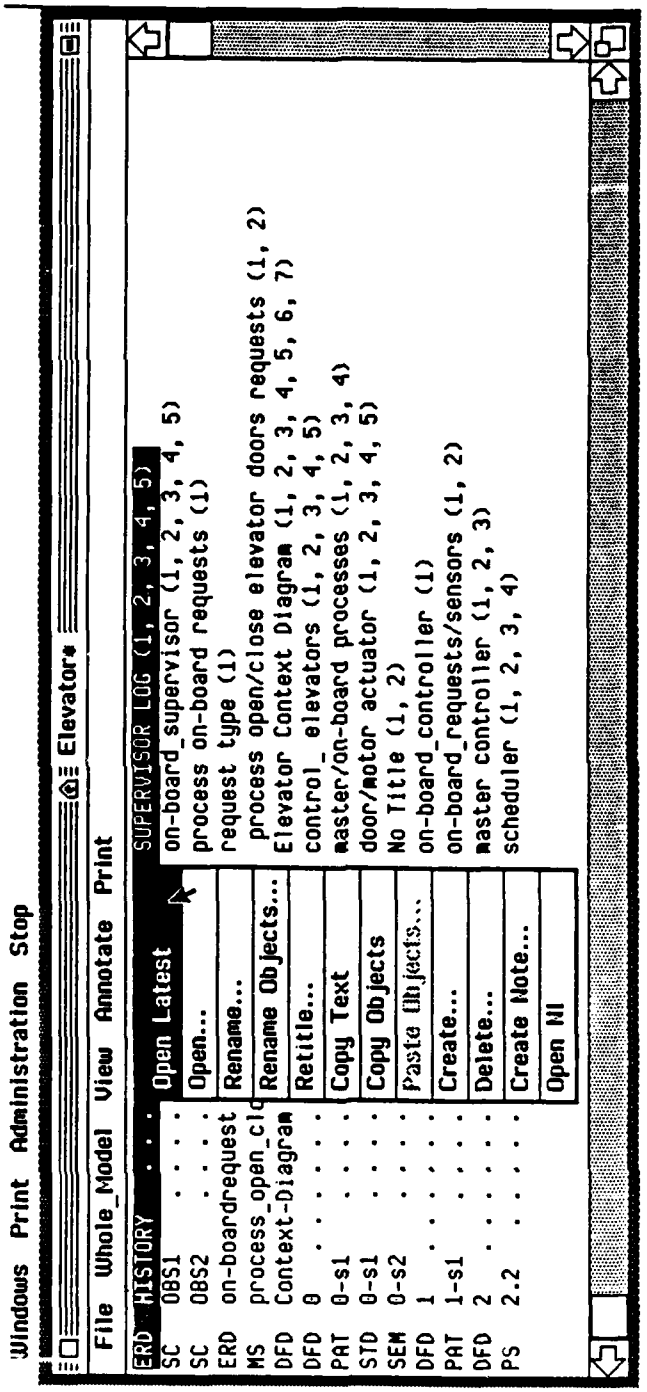


Figure 2 - Process Index for the model of an elevator control system

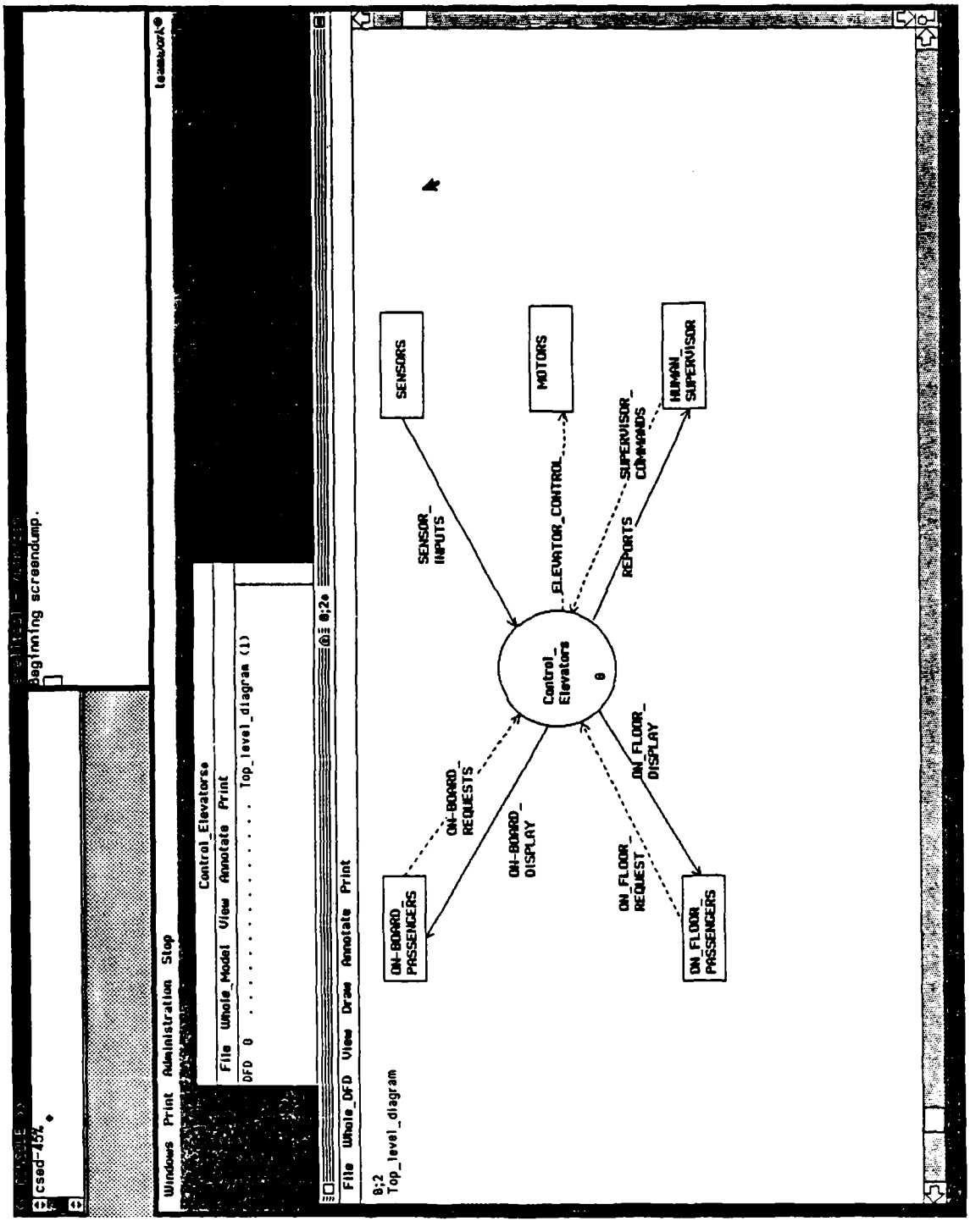


Figure 3 - Data Flow Diagram/Context Diagram for the model

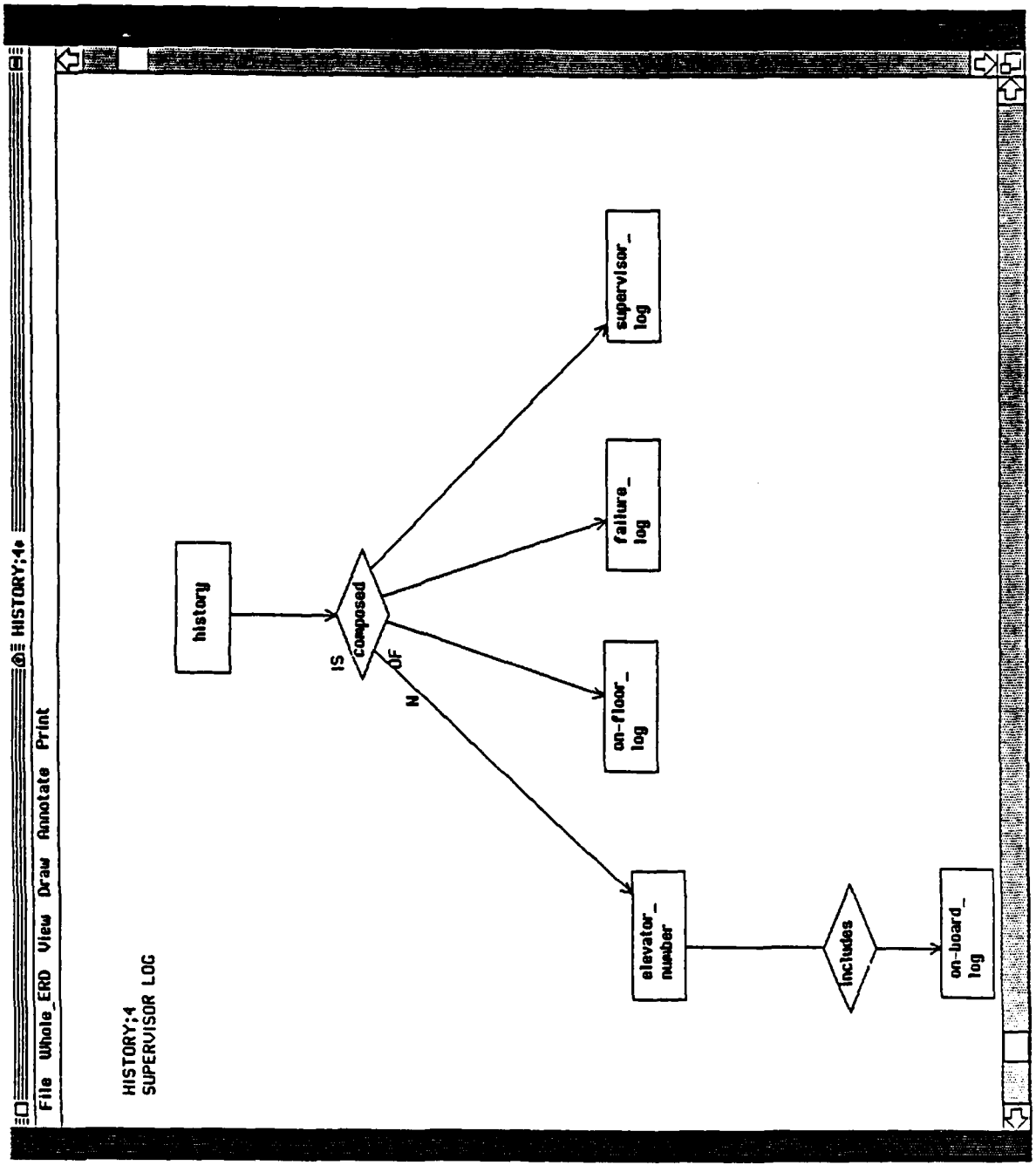


Figure 4 - Entity Relationship Diagram

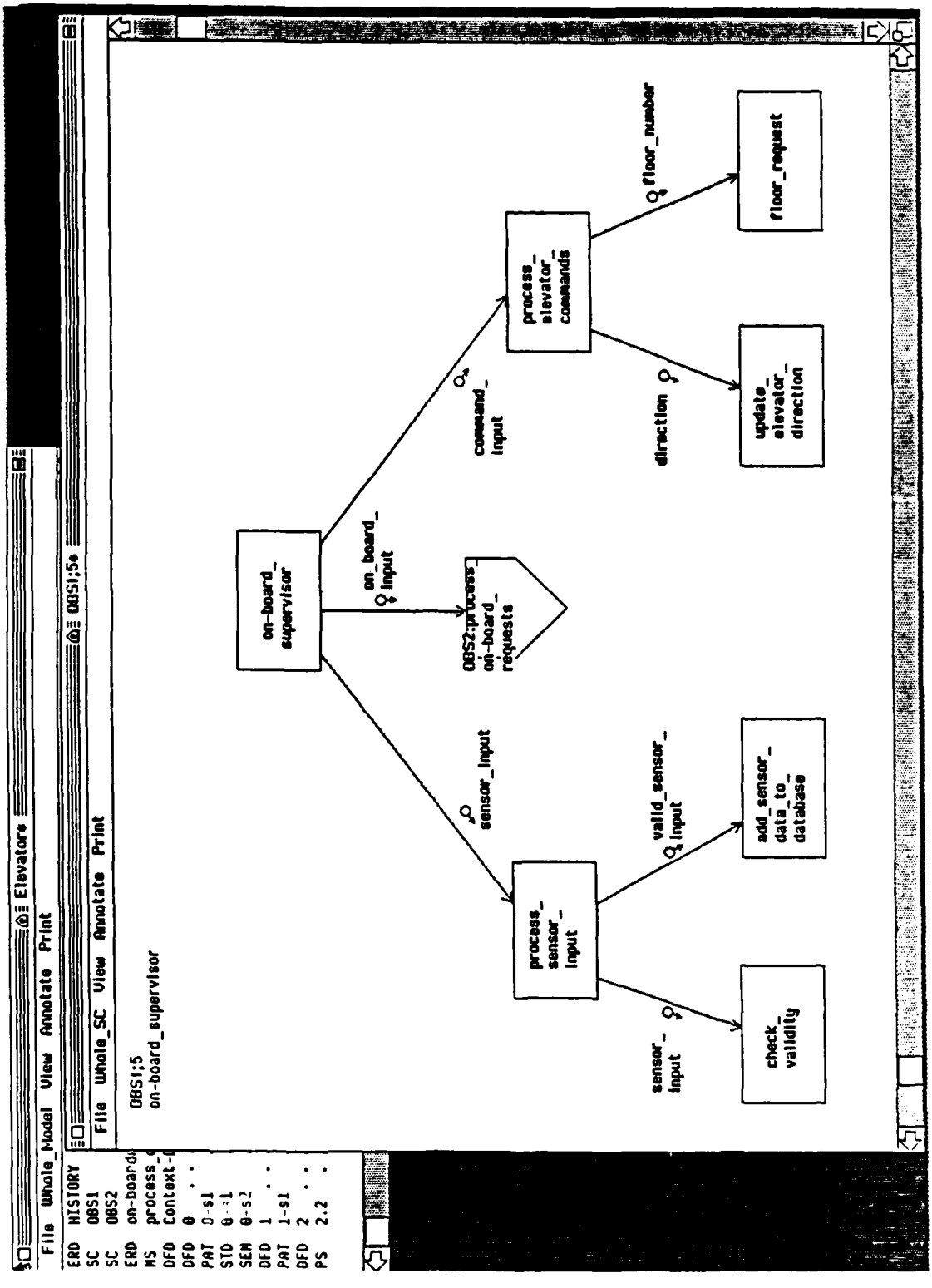


Figure 5 - Structure chart for a process

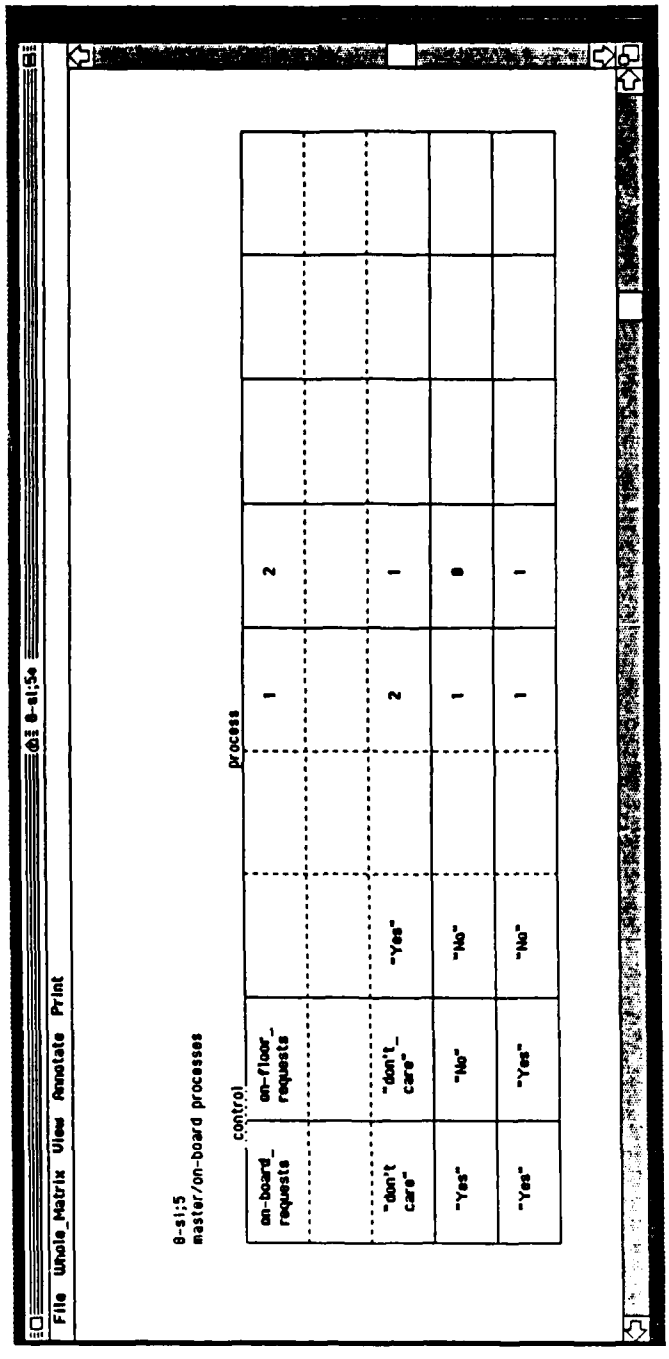


Figure 6 - Process Activation Table

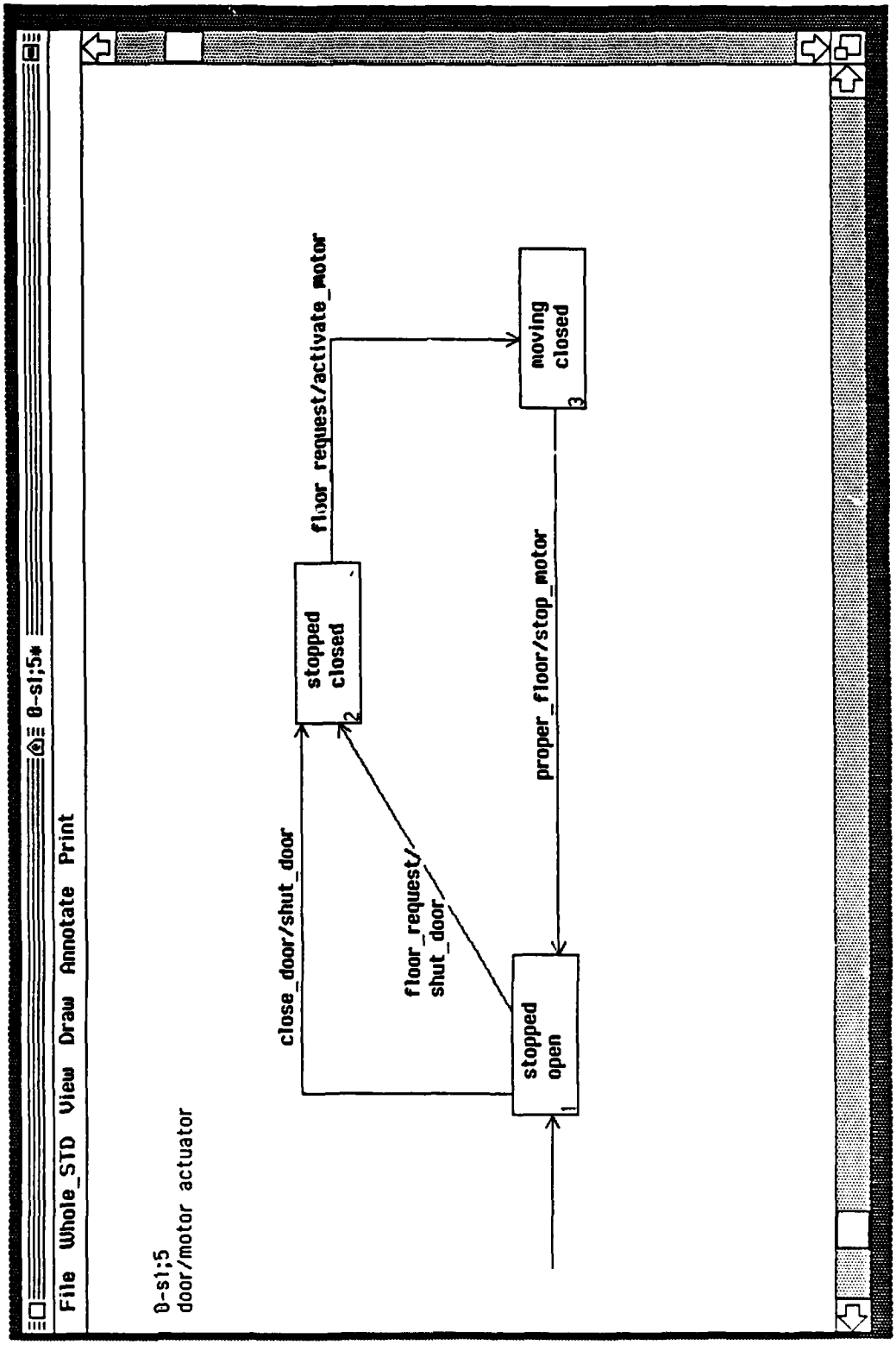


Figure 7 - State Transition Diagram

NAME:
PROCESS_OPEN_CLOSE_DOOR_REQUESTS;2

TITLE:
process open/close elevator doors requests

PARAMETERS:

LOCALS:

BODY:

```
begin

  if elevator_position = on_floor then

    * process command *
    if command = open_door then

      send open door signal

    else

      * must be a close door command *
      send close door signal

    end if

  else

    * between floors *
    cancel request

  end if

end
```

Figure 8 - Module Specification

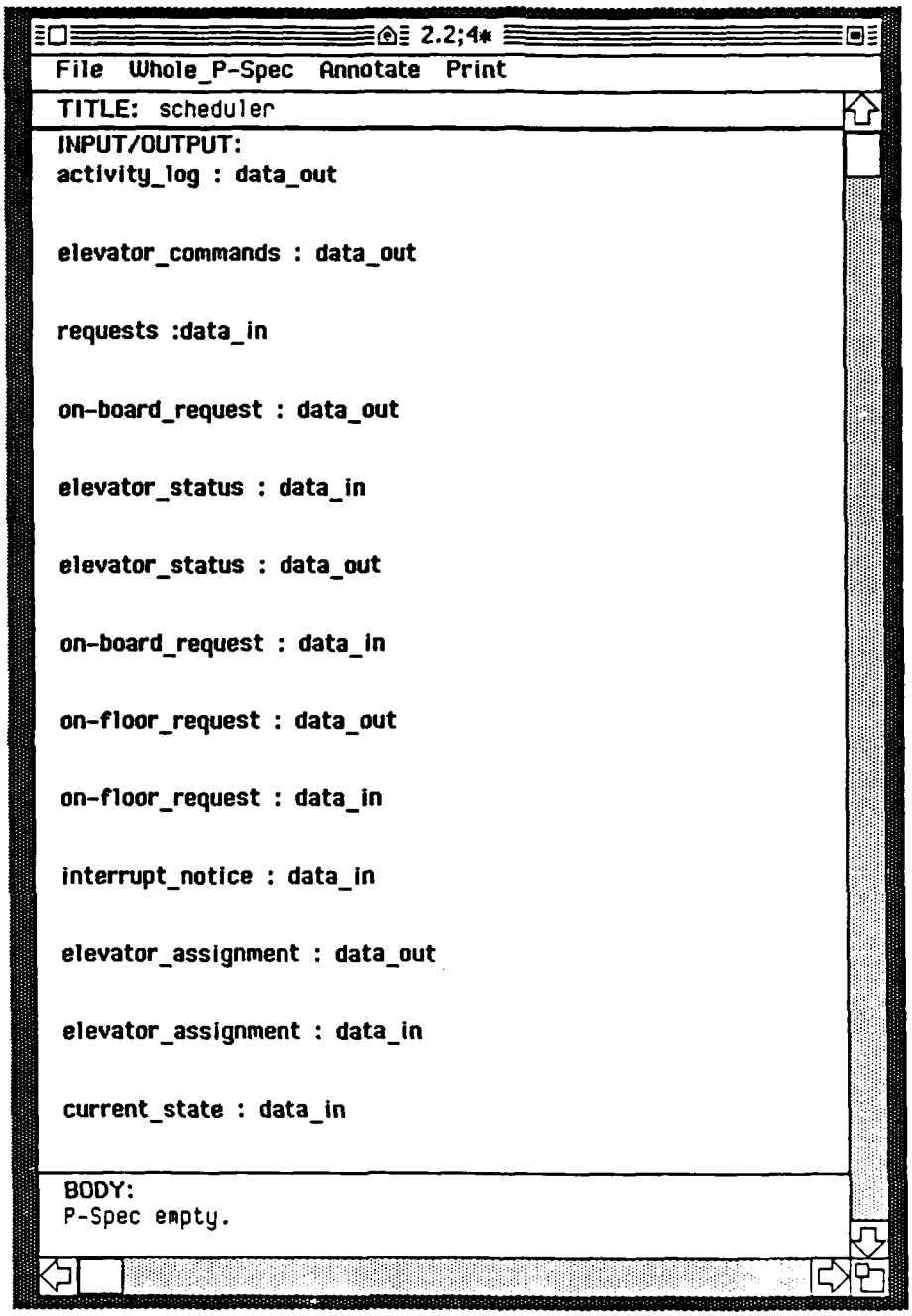


Figure 9 - Process Specification

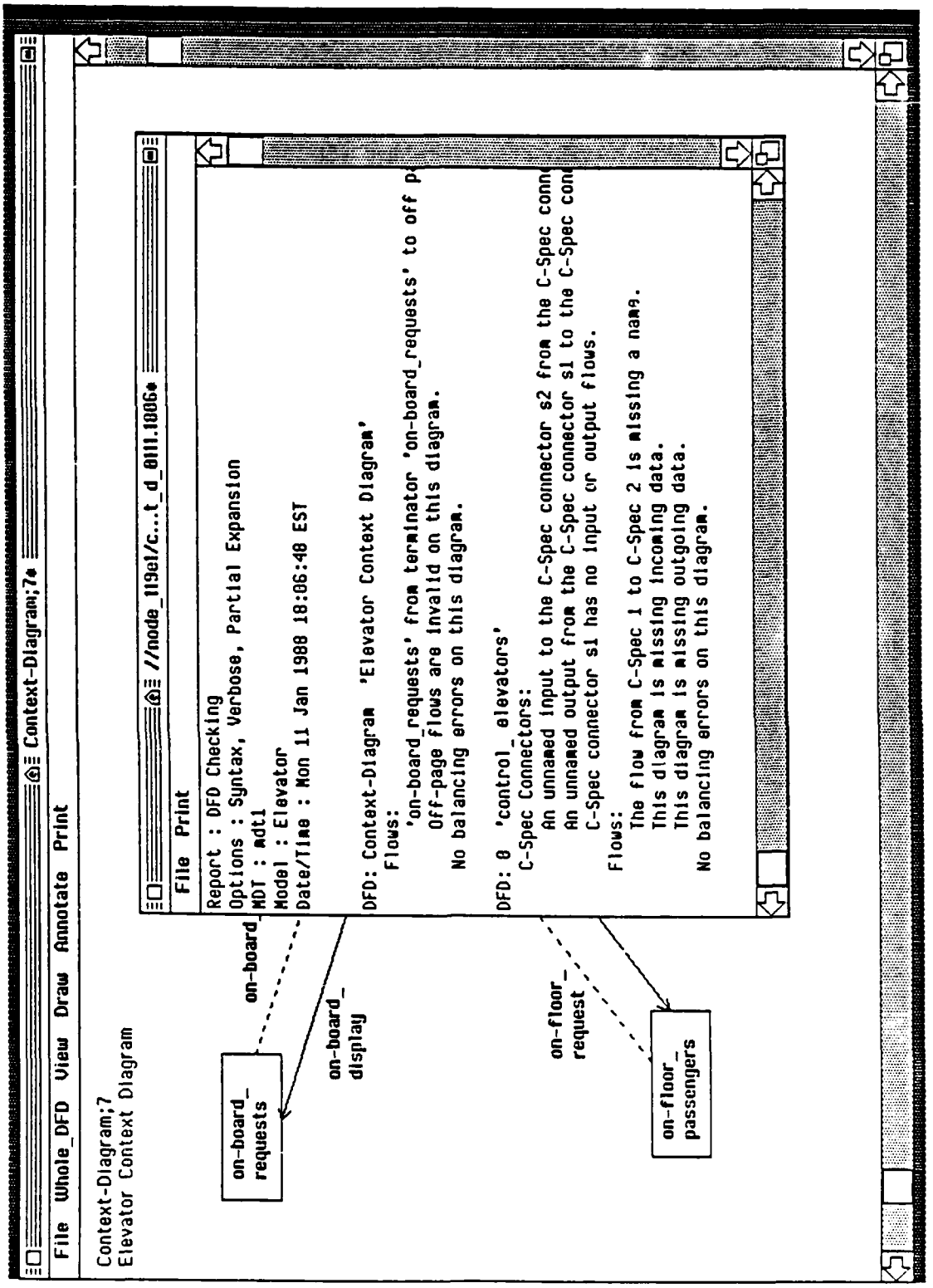


Figure 10 - Sample of consistency checking output

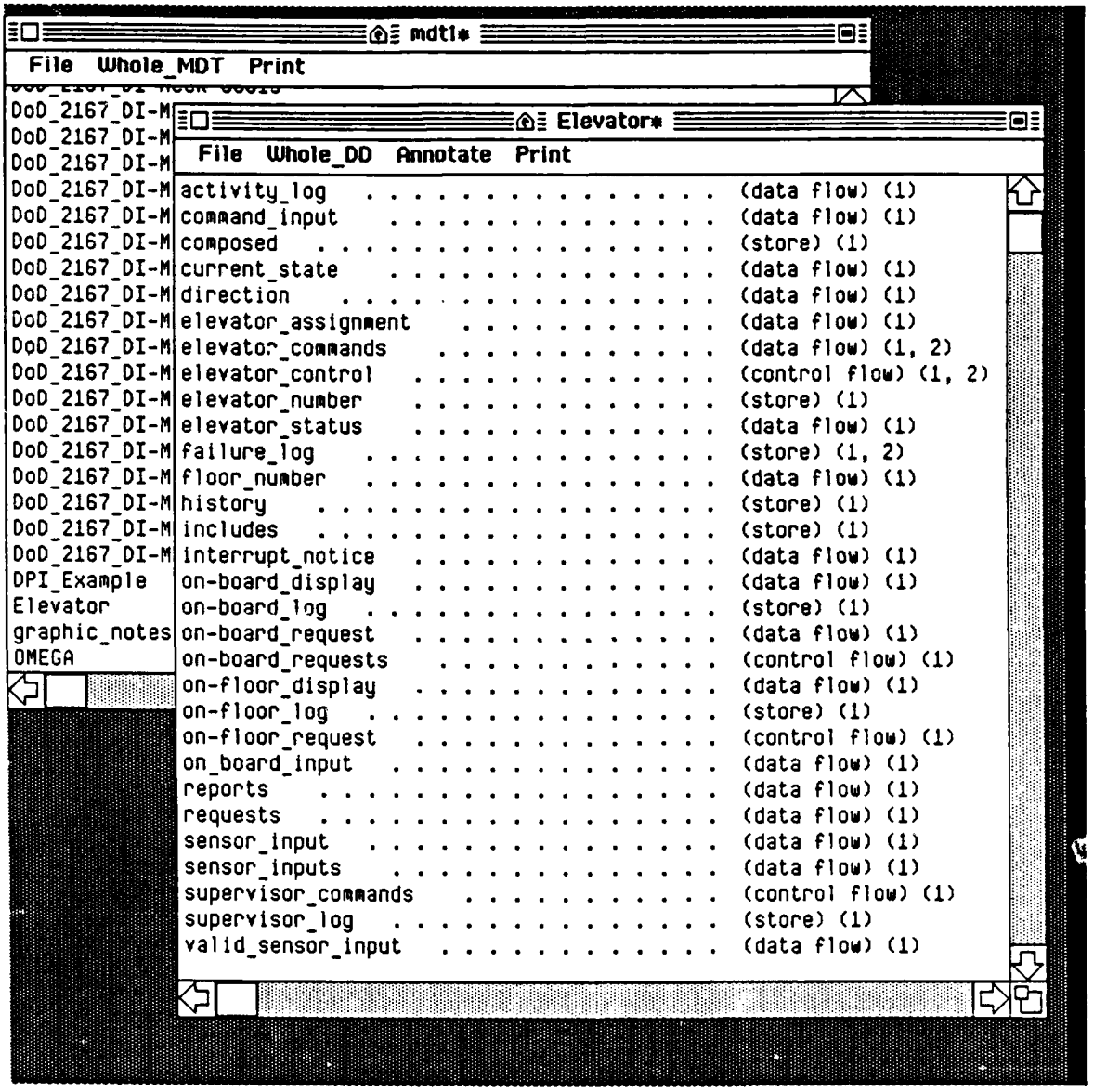


Figure 11 - Data dictionary listing

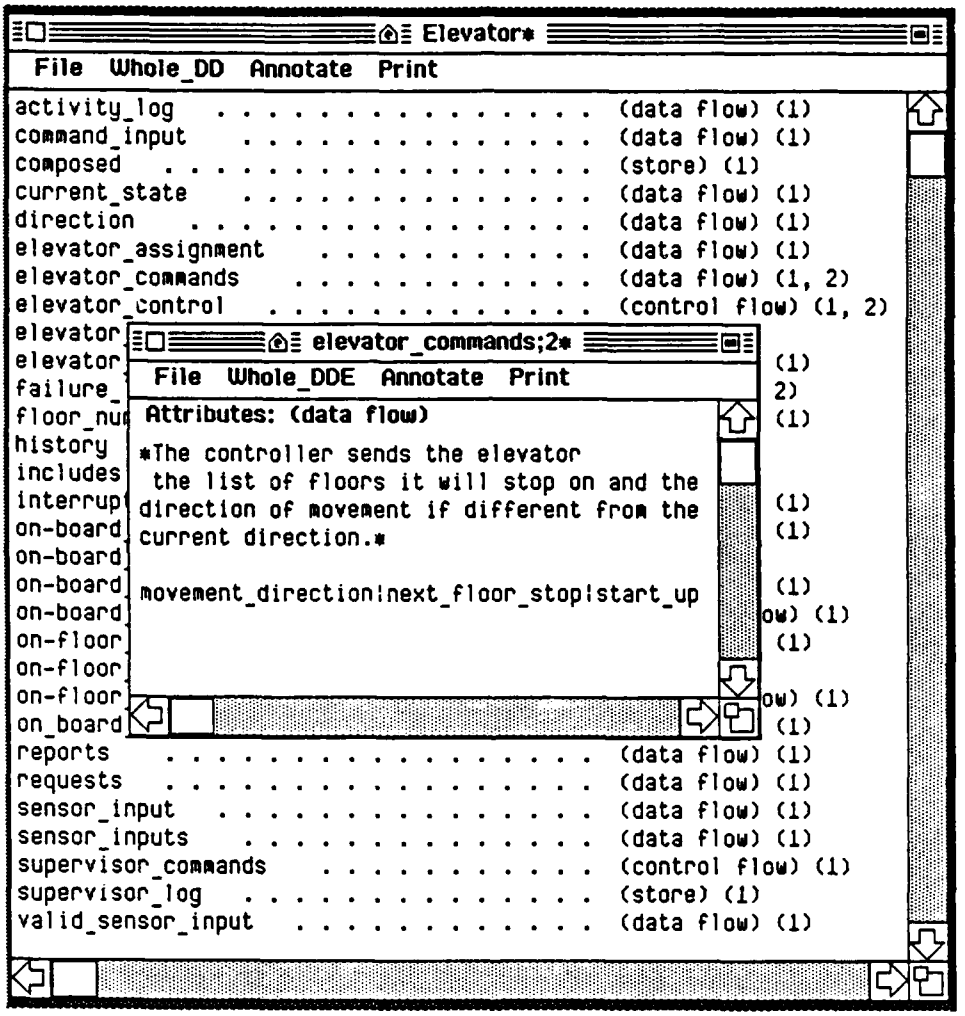


Figure 12 - Sample data dictionary entry

Appendix B
Tool Evaluations

1. SECURITY AND SET-UP PROCEDURES

1.1 Describe any security features the system provides, such as a user name and password, beyond those provided by the host operating system.

TAGS

On entering the TAGS system, via the "TAGS" command, the user identification and password must be given.

AUTO-G

None.

DCDS

The only security feature provided by the tool is a password feature which restricts who is permitted to change the database schema, for example, to add a new type of element, or delete an existing type.

STP

None.

TEAMWORK

None.

1.2 Describe any access permissions the system provides you to share your design work with other individuals or a group.

TAGS

For the design of a large system, the TAGS database would reside on a file-server accessible through a network. The TAGS development environment allows separate components within a system to be assigned to an individual designer or group of designers. Access to the design of these components could then be limited to those individuals.

Synchronization and merging of different designer results is difficult for all tools. In TAGS, it is possible to decompose any block into separate blocks; these may then be copied as separate systems and assigned to different designers. They would work separately. Later, TAGS has features to help merge their efforts (not tried by IDA; the names of the parts may need to be carefully worked out, and the dictionaries administered externally to ease the integration/merging).

AUTO-G

Typically, a system is decomposed into documents (interface and subsystem definitions) by a system administrator, who sets the operating system access permissions for each document. Individual designers restrict their effort to individual documents, for which they are given read/write permission enforced by Unix. Others may be given read permission to view and thus share one designer's results.

DCDS

There are no access permissions provided by the program to share work. It is possible to share work using the SHARE command in the VAX version, but only one person may use a database at a time.

STP

There is a lock-handling facility which permits a designated project administrator to restrict read and write access permissions on particular diagrams or project databases to specified users or user groups.

TEAMWORK

None - anyone who can access Teamwork can read any part of the design, and modify it unless locks are set to prevent it. A capability planned for Version 3 (1988) will allow control on which individuals may open and set locks on design parts.

1.3 How does the system let you know if someone has used the same name to identify a project?

TAGS

When the user attempts to create a system using the name of an extant system, TAGS notifies the user: "SYSTEM ALREADY EXISTS".

AUTO-G

Systems would be catalogued in separate Unix directories, so name conflicts are handled by Unix conventions. The user accesses named "views" of a system. If the name chosen for a view is already being used, then that view would be displayed.

DCDS

DCDS does not let you know if the same project name has already been used. It merely overwrites the current file with the same name on the AT version. On the VAX, DCDS creates a new version of the database.

STP

Once a user has created a project database, any access using that name relates to that system.

TEAMWORK

A project index is provided. A name that is already used will reference the project or system given that name.

1.4 How does the system protect against unauthorized use/viewing of a particular part of a system design?

TAGS

Within the TAGS environment, a user or a group of users can be given restricted access (read/write or read-only) to the group of diagram and tables associated with an SBD component. In TAGS this group is called a document.

AUTO-G

AUTO-G relies on Unix permissions to restrict access to the separate views and documents of a system design.

DCDS

DCDS provides no means to restrict access to separate parts of a system design.

STP

When the locking facility is engaged, unauthorized users are not allowed to use the system's editors and utilities to access the protected files. Without engaging the locking system there is no protection.

TEAMWORK

Teamwork provides read-only and read/write locks to control modifications to the separate parts of a system design. An unlocked object at present can be locked by any authorized user, see question 1.2.

1.5 Describe how you retrieve a specific document or diagram (i.e., querying capabilities).

TAGS

Choose the ACCESS/CREATE option in the main menu, and give the system name. Only those documents allowed by password protect will be available. Then, in edit mode, the DOC, SEC, and PAGE identifiers must be entered. Otherwise, in display mode, the mouse can be used to select and view parent or child descriptions or diagrams.

AUTO-G

The user may name any individual view of a system as a parameter in the "auto new" command that invokes AUTO-G. A view also can be accessed by the READ command from a pop-up menu once AUTO-G is running.

DCDS

The DCDS QUERY facility can be used to display any part of the database textually. Diagrams may also be displayed graphically by entering the graphical editor BROWSE and typing the diagram's name.

STP

Using the diagram editors, a user can specify the diagram to be viewed or edited. For a hierarchy of diagrams (e.g. DFDs) the user may specify any diagram and subsequently retrieve the parent or child diagrams. The user can also "push" an object on a diagram, such as a data flow in a DFD, to call up the appropriate editor which will retrieve the desired description or definition of that object.

The system also provides querying tools to browse or edit the data dictionary directly.

TEAMWORK

The user interacts with the system through series of windows. First an index is retrieved and a project is selected. Then the user can get a process index listing of all the diagrams and data dictionary entries for the project. The appropriate diagram or entry is retrieved by highlighting the name and selecting the appropriate command in a menu. Within a particular diagram, the user may select items and retrieve parent or child diagrams or dictionary entries.

From the main menu, the following steps are necessary to access a specific diagram:

Select	WINDOW
Select	DIRECT ACCESS
Select	PROCESS INDEX
Select	[DIAGRAM NAME]
Select	OPEN LAST

If the user knows the diagram or document name he may bypass the menu scenario and directly enter the name after using two menu selections (i.e. select WINDOW, select EDIT).

1.6 Describe how you get a directory listing of all systems and system components.

TAGS

To get a listing of all the systems accessible by a particular user, he or she can use the SYSTEM - REVIEW/ARCHIVE/LOAD utility. Having invoked this utility the SCROLL UP option will give a listing of these systems. To get a listing of the components of a given system, use the UTILITIES LIBRARY option of the main menu, and the AUDIT option beneath it.

AUTO-G

Typically each system is placed within an exclusive directory, and the operating system is used to list the systems. Within a given directory (system), the database utility, dbutil, will give a listing of all the "views" of the system through the directory command. This can be invoked through a separate Sun window while using AUTO-G.

DCDS

The operating system provides a listing of all systems. A list of all the design components is displayed by invoking QUERY and typing "list all".

STP

They are obtained through use of the data dictionary utilities.

TEAMWORK

The user first retrieves a project index of all systems, then having selected the project of interest, retrieves a process index of all components.

2. USER INTERFACE

2.1 Critique the tool's menus. Are they complete (i.e., do they contain most of the features of the tool that you would access frequently), easy to understand and use, and clearly organized?

TAGS

On the current Apollo version, a fixed menu bar is provided at the top of the display. The SUN version, now being implemented works from pop up windows and pull down menus. The formats are specific to the mode or tool being used and the menus provide easy access to the utilities.

One menu of icons is provided for all diagramming. TAGS needs an intelligent graphics editor to improve this. For example: When one is designing an SBD, only those icons that pertain to SBDs should be presented to the user. In addition, the rubber-banding of connectors between symbols needs to be improved. The TAGS editor should be able to insure the user that "what he or she sees is what he or she gets". It was disconcerting to see what appeared to be two symbols connected, which in fact were not connected -- always revealed to the user by the DIAGNOSTIC ANALYZER or DA. Finally, TAGS should allow the designer to make changes to a diagram or table and analyze the effects of the changes without being forced to write over an existing diagram or table.

AUTO-G

The AUTO-G environment provides a well developed syntax directed graphics editor. The icon menus presented to the user only contain those constructs (e.g., tree nodes) which can validly be added to the G tree at a specified point of construction.

One problem with the current version of the editor is over restriction. Where it would be particularly convenient to "cut and paste" a portion of a G tree, the G-editor in certain cases will not allow it, even though the resulting tree would be syntactically correct.

Besides the menu of available graphic icons, a two level hierarchy of walking menus is available for selecting various viewing, editing, and activity commands.

DCDS

The available graphics editor, BROWSE, is menu-driven. These menus are fairly complete. Key shortcomings are the lack of options to cut and paste diagram elements. Also, the panning facility requires turning off the main menu first, doing a series of panning steps, then reenabling the main menu. ENTRY, QUERY, and other editors are command-driven with no menus or help screens. Only the IBM-PC AT version provides for renaming diagrams.

STP

Each editor or tool has a fixed menu, as well as two level walking menus. The menus are easy to use, clearly organized, and apparently complete.

TEAMWORK

The editor for each diagram type or text has a menu which provides for selecting the common operations. The selectable items are highlighted. Menus are apparently complete; however, in some instances the user is forced to go back to the process index (in 3 step menu selection) or elsewhere to retrieve information that should have been accessible by selecting a menu item.

2.2 Is the method for selecting, pointing, and keying menu options convenient for the user during common command, editing, and display actions?

TAGS

The mouse and keyboard are the only input devices. Mouse selection and use of menu options is rapid and efficient.

AUTO-G

The mouse and keyboard are the only input devices. The icon menu is easy to use, as such, but the icons available in some cases are numerous and so similar that it may be difficult to determine which icon is desired. However, an icon resulting from an incorrect selection can easily be deleted.

DCDS

Only the graphics editor, BROWSE, has menus. In the VAX version, cursor keys are used for selecting and pointing. This requires repetitive keypresses. This is particularly unfortunate since the single cursor must be repeatedly moved back and forth between the menu and diagram areas. (The menu is at a fixed side of the screen.) In the AT version, the mouse provides more convenient pointing and selecting.

STP

Yes, the selection method for using the mouse and pop-up overlaying menus are convenient. However, there are some minor problems. For example, sometimes when windows are overlaid the text identifying the use of a particular window is obscured, making it difficult for a user to determine which window is which.

TEAMWORK

Yes, the mouse and menu interaction works well. Teamwork has context-specific menus that show the choices available for picked objects. These change automatically as you move to different windows with different picked objects. A minor complaint is that on some actions the user must select the menu item of interest and then additionally select "OK" on the same menu.

Several features of the menu system are very convenient. Specifically the ability to move and resize windows or to shrink a window to a one line place holder are very useful.

2.3 Is a mouse required to use the package? If so, describe its use.

TAGS

The mouse is not essential. The cursor keys and other key sequences can perform the same functions as the mouse. The mouse is a selecting or pointing device.

AUTO-G

The mouse is essential for picking diagram elements and icons during editing. Key sequences may be used in lieu of pop-up menu selections.

DCDS

The mouse supported by the PC-AT version is used to point to command choices and objects in the graphical editor - but cursor keys may be used also. The VAX version of DCDS does support the use of a mouse. The cursor keys (i.e., the up, down, left, and right arrow keys) are used to select menu items, edit a diagram, or pan across the working area. By depressing the shift key and an arrow key, the cursor will move in small increments across the diagram.

STP

A mouse is required for choosing menu options and for using the graphical editors. It is used to point to graphical objects and menu choices. The mouse has three buttons. The buttons are pressed separately to select, cancel, or delete, or pull up a menu or window.

TEAMWORK

The mouse is essential. It is used to select an item, call up menus and cancel commands.

2.4 Does the system support a high resolution display yielding high readability while displaying a large, fully labelled design diagram?

TAGS

Yes, the Apollo provides a 17" diagonal high resolution display. The TAGS method constrains diagrams into pages (with connectors to join flows onto other pages).

AUTO-G

The Sun workstation provides a 19" diagonal high-resolution display. AUTO-G provides a "window" into the view being displayed, which gives a readable, but possibly small part of it.

DCDS

No, the current system does not support a high resolution display. Current implementations run on the VAX supporting the Tektronics graphics terminal, or on an EGA board for the AT. Neither implementation allows fully labelled designs, but only 8 characters per object.

STP

Yes, on the Sun workstation used by IDA.

TEAMWORK

Yes, Teamwork is ported to many workstations which have high resolution displays, as listed in Appendix A.

2.5 Describe any problems you had in specifying the design using the graphics editor. For example: Was it difficult to add or delete text, or place an object in a certain location in a diagram?

TAGS

The major problems in the use of the graphics editor are:

- The system does not "rubber band" the arcs between blocks or flow elements. This means that the user must guide the line around other objects and may later have to delete and redraw it when a need to reroute it occurs. Moving a block may also "break" a connection (arc).
- Because the graphics editor is not syntax directed, it is possible to terminate the line a very short distance away from the block to which it should be connected. This is only later discovered (during the DA, where a lack of connectivity is found).
- The name of the connection (arc) between two blocks in an SBD is always the same as the DOC-name with a hyphen and number; e.g., INFO_SYSTEM-12. Since these are required prefix names, it would be reasonable to provide them for additional editing.
- There is no symbolic text editor. In several cases, a single change was to be replicated in many places, and an edit facility would have been very useful.
- Although a symbol or group of symbols may be copied from one diagram to itself, another diagram, or many others, there may be several copies on top of one another (and found either by bringing up the LABEL on the symbols, where two different numbers are superimposed, or else when running the DA and finding unexpected errors).
- Error and other system messages appear in a rather small area at the bottom of the screen. Since the system messages are routine (such as SAVED), the errors may be ignored. Also, in some cases, the error message number was not recorded in the manual.
- There is no way to highlight or query a diagram for the component number given in an error message. If there are many components, scanning the diagram for the error is time consuming.

AUTO-G

AUTO-G provides a easily used graphics editor for creating and editing G-trees. For large and fully revealed diagrams, the display provides a small viewing window on the whole. Inevitably, when the designer decomposes a process (tree node) into subprocesses, which are placed subordinate, they may not fit within the viewing window. Experience and discipline in using G should eliminate this as a

problem. Any object on a diagram can be hidden (removed), leaving only a small residual marking (a few periods or dashes) to show its existence. Easy zooming, down to nine discrete reduction levels, is provided, but only four seem useful. Readability is good at the two highest zoom levels where text labels are presented. Picking a desired object precisely can be troublesome also, since diagrams carry many specification elements and many are close together spatially. One or more diagram objects may be marked, to speed up otherwise repetitive operations, such as delete, move, or copy.

DCDS

Problems include:

- When an element is deleted, all elements below the object are also deleted. This is a problem due to the lack of cut and paste facilities, a renaming capability on the VAX, and because all diagrams have a hierarchial structure.

- Diagram editing is slow and tedious. Editing a diagram requires repetitive keystrokes to move to a menu window, choose an option, and then repetitive keystrokes to move back to an object.

- The graphics editor provides no control over the layout of the picture. For instance, you cannot specify which side of the parent node a child node will be placed. This makes the starting point of the diagram rapidly extend off the screen. If the diagram is large, the user then must use many additional keystrokes to pan across the screen to the starting point of the model in the VAX version. Moreover, in the VAX version, when any menu option is chosen, the diagram is redrawn in its original position, requiring the user to re-pan the diagram.

- Graphical objects which require extensions, such as the parallel symbol, are not automatically extended, and the graphics editor allows them to be incorrectly extended.

- Syntactically incomplete or incorrect diagrams cannot be saved. Furthermore, there are no error diagnostic messages.

STP

The following problems were encountered:

- The Ada or C code generated for data definitions of enumerated types is not correct.

- Any symbols which are overlaid by a symbol of the same type disappear. This is particularly important when two or more arcs for data flows connect the same objects, for unless the user introduces extra vertices to force the flows into different arcs, both arcs disappear from the screen. If for some

reason the invisible objects become unconnected, or errors are reported during checking, the symbols cannot be located graphically for correction. Similarly, anchors are too small in the diagrams. If an anchor becomes unconnected it may easily be forgotten until the checking facilities report an error. It is then difficult to spot and correct. The system should point to an object giving the error. Problems can also arise due to the similarity between anchors and vertices.

- The same diagram can be shown in more than one window, and updated differently in each window if file locking is not engaged.

- The existence of decomposition or definition of a DFD process is indicated by an "*" or "p." in the process bubble. There are no such notations to indicate decomposition or definition of any other elements in DFDs, or in other diagram types.

- When decomposing an item for the first time using the push command, the new graphical decomposition is in an extremely difficult-to-use format. It takes some time to move parts around into a usable format.

- The pipetool window always displays a message indicating the task it is performing or whether it is waiting for the next command. Windows used for editors should also do this. When certain commands are requested (e.g., invoking a new editor) there can be a long wait, during which the user is not sure what, if anything, is happening.

- When processes in a DFD are deleted, a menu option is provided to allow changing process indices. Once changed, the new process index value is supposed to be remembered so any new processes are indexed correctly. This did not work.

- The system permits a user to decompose a non-leaf module in a structure chart. Also a data parameter pertaining to a particular arc between modules can be positioned anywhere in the diagram, even beside a different arc.

- The system permits a user to define more than one process on the context diagram. This leads to various problems when all those initial processes are decomposed. The tool relies on numbering with process 0 as the context diagram, processes 1, 2, 3, ..., as the first level decomposition, processes 1.1, 1.2, 1.3, ..., as the decomposition of process 1, and so on.

- There is no way to type in the name of an object and have the system skip directly to that object. This makes large systems more difficult to use, since finding an object's decomposition may require popping through a number of levels.

- In a data structure diagram, when two or more selection boxes are placed in close proximity, the selection symbol "o" jumps out of the boxes to lie between them. Actually, the "o" symbol remains in the box, but the

overlapping symbols give the appearance described here, since the lines cancel one another.

- If the database is closed during the creation or modification of a diagram, the new diagram is stored two-levels up in the directory. However, the user is not informed of this.

- The tool does not provide for "popping" back to the data flow diagram from the data structure and entity relationship editors.

- When the user chooses the "push" option from the menu, the submenu of options for decomposing or defining a process does not always appear.

TEAMWORK

Some problems were encountered:

- Text can be overlayed (actually it is not exactly aligned on the vertical axis) on other text with little indication that this has happened.

- During the course of a graphics edit, the screen usually had to be redrawn multiple times due to lines or text missing. Version 2.3 display enhancements are intended to eliminate this through automatic repainting.

- Text cannot be selected on a diagram when using less that a 1:1 scaling factor.

- With a 1:1 factor, picking is touchy, and repeated tries are necessary to successfully pick the desired one.

- An object opened in another window for update is locked and cannot be updated until the user finds the window and closes it; Teamwork version 2.2 does not indicate who has the object open, but version 2.3 will remedy this problem. A facility to automatically retrieve the conflicting window with the name of the user who has it opened is desirable.

- Consistency between the graphics and data dictionary entries is not automatically maintained. For instance, if an object's name is changed in the data dictionary all the diagrams which contain the name must be manually edited to reflect the change and vice versa. Changing the name of an object is therefore difficult. A feature which assists such changes is the "where referenced" choice invoked from the data dictionary index. It will show all places that an original name is referenced in diagrams and tables.

2.6 Does the system provide undo capability?

TAGS

No.

AUTO-G

No.

DCDS

No.

STP

Yes, the one-level undo key lets an inexperienced user experiment while knowing he can undo an action if necessary.

TEAMWORK

If a mistake was made the user could recover by either undoing the modifications or, if the diagram had been saved, by rolling back to a prior version. The UNDO feature and its opposite function REDO, were particularly nice as the user could step backwards through diagram changes for multiple steps rather than being limited to undoing the last action only. Unlimited UNDO is provided in every window.

2.7 Does the tool provide an acceptably quick response to designer actions and output requirements?

TAGS

Most of the time-consuming operations (e.g. analysis and printing) are spun-off as background processes. A mechanism for checking the status of these processes is provided. The response for editing and display actions is good.

AUTO-G

The G-editor is relatively slow when adding to or deleting from large diagrams. The semantic analyzer SEMA seems to function in a timely fashion.

DCDS

The graphics editor, BROWSE, does not provide as quick a response to user requirements as QUERY. In particular, on the VAX version, it takes about 10 seconds to redraw the screen after any change to a diagram. QUERY does provide acceptably quick responses to user requests.

STP

The tool does give a quick response to editors commands providing it does not require a change of diagram editors. However, if a new editor is required, there is a noticeable delay, of up to about 30 seconds on the Sun 3/50 that IDA used..

TEAMWORK

The most frequent activities have very good response time. Consistency checking and the initial Teamwork login are noticeably slow. Consistency checking of multiple levels of a large model could become time consuming. Two ways Teamwork helps reduce the impact are: do subtree checking, from one decomposition level downward, rather than the entire design; and, do entire design checking as a background process (available in Version 2.3).

2.8 Cite the worst case you encountered of slow response or lengthy set-up to complete an action?

TAGS

Slow response may be encountered in: 1) printing; 2) diagnostic analysis; 3) simulation (not exercised by IDA).

AUTO-G

Plotting a G-tree on the Hewlett Packard 7475 plotter is rather slow, as would be expected, and should be done as a Unix background process to eliminate this problem. A major deletion or addition to a diagram may provoke a rather slow (15-30 second) response on the Sun workstation.

DCDS

The worst case is loading and saving the database, which on the AT version can take 1 to 2 minutes for a small database, using a Bernoulli disk. The VAX version requires about 1 minute to load the database, depending on the current workload of the machine.

STP

The worst case for the tool is starting up, with times of up to 5 minutes common. The system was also slow in changing applications, up to 30 seconds per change.

TEAMWORK

The worst case response time for some activities is highly dependent upon the size of the project model. For our small elevator model and the medium size Cruise Control system provided with Teamwork the consistency check time was 15-45 seconds on the Apollo 3000. The amount of time required to bring up Teamwork is about 50 seconds. Switching editors and diagrams was fast with response times of 1-5 seconds observed. Diagrams which have already been read in are evidently buffered by Teamwork or the underlying DBMS since their retrieval time is about 1 second.

2.9 Do text or table editors provide templates or standard formats for entering and editing data?

TAGS

TAGS provides tables with predefined columns for defining variables and data types.

AUTO-G

G is a highly graphic language, so that required text entries are minimal and consist of short names, labels, expressions, and comments. AUTO-G provides the user with a single entry line at the top of the editor window in which to enter the required name or text.

DCDS

BROWSE in its textual form provides templates for entering and editing data. DCDS does have a synonym capability.

STP

In the case of data dictionary items, there are a number of standard templates which may be modified by the user. The data entry format is consistent throughout. Abbreviations are not allowed. Synonyms are provided by the ALIAS feature. The user must be familiar with a text editor available on the host operating system.

TEAMWORK

The editor commands are consistent between different window types. Version 2.2 allows user-defined data dictionary templates. In Version 2.3, users can predefine their own templates for process and module specifications. In Version 3, more enhancement toward user defined menus and forms is planned.

2.10 Does the tool automatically number and label processes and data flows where previous information or conventions make this possible, or does the tool force the user to number and label all processes and flows?

TAGS

In TAGS, the designer explicitly defines the data flow between components and names and numbers each component and connector. There is little or no aid in specifying system decomposition, except that the DA will show errors from incorrect operation. This aspect is expected to be improved, but Teledyne Brown has no delivery date yet.

AUTO-G

No, all process elements and signals are labelled by the designer. In fact, signal and message declarations must be propagated to the superior levels in a tree, which can require many tedious entries.

DCDS

There is no automatic numbering or labelling. Everything must be explicitly named by the user.

STP

The tool automatically indexes processes and external entities used in DFDs. Processes may be reindexed, but the tool provides no way to reindex external entities. No other DFD elements, or elements used in other diagrams are labeled automatically. When a process is decomposed at a lower level, the external flow labels are automatically provided.

TEAMWORK

Teamwork automatically numbers processes on DFDs by the Yourdon-DeMarco numbering scheme. Teamwork provides facilities to renumber processes on diagrams if the user considers this necessary.

2.11 Is it easy to pan across the diagram viewing area?

TAGS

TAGS enforces a discipline upon the user of dividing diagrams and tables into pages. Each page is then viewed separately. Panning is not required.

AUTO-G

To scroll across a document, one selects a node within the G-tree and then repositions it at the opposite side of the screen, thereby repositioning the G-tree, a time consuming process. The AUTO-G environment provides the user the ability to scale G-trees, to allow more of the tree to fit on the screen at any moment. The problem in dealing with a scaled down view is that all the text disappears. Unless the user is familiar enough with the particular G-tree, he will be unable to navigate about the tree. Additionally, for all practical purposes, editing is limited to trees at full scale.

DCDS

Panning across the viewing area takes considerable time since every time the user steps across the drawing area, a complete redrawing of the diagram and menu occurs.

STP

STP provides scroll bars for panning across diagrams. STP also provides a panning area in all editors other than PICture.

TEAMWORK

Yes; Teamwork provides scroll bars to pan across the viewing area.

2.12 Describe the zooming and/or windowing capabilities.

TAGS

No zoom-in capability is needed, given its single page diagramming constraints.

AUTO-G

AUTO-G provides the designer with nine zoom-out levels ("scales"), to view a G-tree as an unlabeled miniature of the original unscaled tree. Using a scaled view of the tree, the designer can select a tree node for centering the unscaled view. This provides an alternative means to scroll the diagram.

DCDS

There is no windowing capability. The evaluators were unable to find a limit to the number of zooming levels. The text contained in the diagrams became illegible when the zooming level was either very high or very low.

STP

The windowing capabilities are excellent, as they fully utilize Sun's windowing system. Zooming is easy to do, but only three zooming sizes are allowed.

TEAMWORK

Four zoom sizes are provided by Teamwork: 1:1, 1:2, 1:4, and 1:8. A diagram with a zoom setting of 1:4 and below is very difficult to read. We felt that a 3:4 or user selectable zoom setting would be useful. Windowing capabilities are good, allowing for the repositioning and sizing of each window and even shrinking it to a title bar.

2.13 Can you make entries in the data dictionary while using the graphics editor, without having to switch editors?

TAGS

All dictionary inputs are done through one separate editor, in terms of the parameter tables.

AUTO-G

Yes, because the graphics editor is used for all entries, including text labels, variable declarations, and annotations.

DCDS

No, ENTRY, the textual input editor, would be used.

STP

Yes. While in a graphical editor, the user can select the option "Generate data dictionary," and all data dictionary entries are made automatically. (Note that there is no update data dictionary option, the dictionary is generated each time.) Short textual annotations which can be associated with diagram element are not stored in the data dictionary.

TEAMWORK

Yes, in the graphical editor the user can create and update data dictionary entries for objects in the window. Some data dictionary edits however can only be performed by entering a data dictionary element through the data dictionary editor.

In addition if an element name change is made the corresponding elements on diagrams are not updated.

Version 2.3 of Teamwork supports free hand graphics to enhance the ability to annotate diagrams or objects. In addition, Teamwork/DPI is available, that allows the merging of graphics and text from different sources onto the same page during document preparation.

2.14 How much time (in working hours) did you spend in training and familiarization before beginning the first design project?

TAGS

About 3 days of training and some practice, plus a subsequent week of relatively slow practice.

AUTO-G

About 5 days of training and practice with two instructors.

DCDS

The evaluators spent eight hours in training before starting the first design exercise, and spent 14 hours more in training before starting the next design exercise.

STP

Two days.

TEAMWORK

Approximately 2 days; the system is relatively easy to use.

2.15 Describe how easy it was for you, with the training provided, to learn to use the basic system features quickly and efficiently?

TAGS

As with any system of any complexity, one is constantly learning new aspects of the system up to and including the very last day. TAGS is easy to become familiar with to allow a novice to use it effectively within 5 days. IDA took most advantage of calls on the HOT-LINE after training and a week's hand-holding by Teledyne Brown.

AUTO-G

It was no more difficult than tools such as TAGS or DCDS, that are comparable in depth of specification.

DCDS

It took 4-5 hours to feel confident with the basic SSL concepts, but it took another two more hours to feel comfortable with the syntax of DCDS's ENTRY tool.

STP

By and large, this is an easy tool to learn. The two chief problems found were: 1) when a diagram reached a certain size/scale, all text was suddenly hidden; some hours were lost identifying what had happened. 2) it is supposedly possible to change the default restricting external entities to the DFD context diagrams. Many hours were lost finding that this was not so.

TEAMWORK

It was easy to learn the system. Some improvements in the area of new user support would be welcome. There is no online help facility. Also, the documentation does not thoroughly discuss the design methodologies and the relationship of the various tools to one another.

2.16 Describe what online help is available.

TAGS

There is an on line help facility for the keyboard layout, which is particularly useful for the MATH keyboard option -- it would be tough to remember all the math symbols provided by TAGS (e.g. an upper-case 7 for Greek iota). There is no tutorial online.

AUTO-G

There is on-line help indicated for the dbutil (database utility) facility; although for the IDA evaluation, AUTO-G was unable to access it. There is no tutorial on-line.

DCDS

There is no online help available.

STP

There is a help mode which gives terse messages, enough to assist a user who is familiar with the package. There are no detailed levels of online help. The help messages are user-customizable and are accessed via IDEtool.

TEAMWORK

None. Most users with regular experience on Teamwork apparently find it easy enough to recall the commands or choices.

2.17 Does the system provide user selective tailoring of basic graphic characteristics, such as line width, text size, icon relative size, and diagram element placement?

TAGS

Diagram elements may be placed anywhere, and icon size automatically adjusted to accommodate contained text.

AUTO-G

Icons are automatically sized to enclose text, and diagrams drawn to eliminate text overlap.

DCDS

No.

STP

Diagram elements may be placed anywhere on the graphics display. There is no control of line width or text size. Icon size can be adjusted to accommodate contained text.

TEAMWORK

Teamwork allows the user to place diagram elements anywhere on the graphics display and to size icons to enclose their text. Teamwork does not have the ability to tailor line size, width, or relative size of text.

3. DESIGN SEMANTICS

3.1 Describe a representative set of the symbols or icons provided for describing a system, versus use of text and labels?

TAGS

Symbols other than connecting lines include:

- Rectangles, for specifying an expression or series of expressions or for specifying a component in a System Block Diagram
- Conditionals (hot-dog shaped), with two exits
- Blocks with two additional vertical lines for specifying a procedure
- START, ENTRY, and terminate symbols for IORTDs and PPDs, etc.
- Hexagons and curve-sided boxes, for input/output communication and transmission (including delays)
- Combinatorial symbols (circles with various markings) used to construct paths connecting various I/O events and processing symbols, i.e., Fan-Out and Fan-In symbols
- Comments, using a triangle to start a note
- Labels to boxes, connecting lines, and other required text
- External (dashed) and internal (solid) connector lines
- Terminator symbol for end of logic thread.

AUTO-G

The G language uses rectangle as the basis for most of the constructs in the language. The rectangle is embellished with additional marks to symbolize a unique function or meaning. For example:

- A square loop in the upper right hand corner of a rectangle indicates the structure (represented by the rectangle) is repeated, i.e., an array whose elements are member of that structure
- A solid box centered along the top edge of a rectangle indicates the structure is a template, i.e. a parameterizable generic representation
- A wedge entering (leaving) the side of a rectangle indicates that the structure (process) is receiving (sending) a signal
- A (solid) diamond centered on the upper left hand corner of a rectangle indicates a function template (reference)

- A diamond centered along the side of a rectangle indicates the structure represents a data object (variable) which may be read and write, read only, or write only storage, depending on whether the diamond is solid, or half solid and if half solid which half
- A hot dog shape signifies a state of a process.

DCDS

F_nets are the diagram form for describing a system. At this level, requirements are specified in terms of general functions to be performed and their order of execution. Any system description can consist of multiple layers, which are usually standard processing, exception and resource handling layers. The same symbols are used for describing each layer. These symbols are function, and parallel, replicate, selection, and iteration of functions. There are also symbols for triggering and clearing of exceptions between layers, and for asserting and checking of data items.

DCDS uses Requirement_Networks (R_nets) to specify software requirements. An R_net consists of an input stimulus which starts the processing of the R_net and a flowchart-like diagram. Each R_net can have various validation points. Timing requirements for paths between the validation points can also be stored in the database.

DCDS uses DDL to specify hardware requirements and the mapping of software requirements to processes and hardware. At this level the rates of buses, links, devices and processors may be specified.

STP

The choice of symbols depends on the editor used and the methodology that editor supports. The different diagram types provided are Data Flow Diagrams (DFDs), Entity Relationship Diagrams (ERDs), State Transition Diagrams (STDs), Structure Charts and Data Structure Diagrams. For DFDs, either Yourdon - DeMarco or Gane-Sarson icons may be used.

A Picture Editor provides an additional set of symbols that can be used in developing a user-defined type of diagram. However, the user would have to modify the underlying control information file used by the system, and possibly pieces of code, to integrate any such diagrams with the predefined ones.

TEAMWORK

Standard symbols are defined for the data and control flow diagrams (DeMarco's symbols), and for structure charts, entity relationship diagrams, state transition diagrams, process activation tables, and state/event tables.

3.2 Does the system support timing and performance specification?

TAGS

Timing must be defined for input and output events. Time tags can be set by input and output events, that permit checking of relative or elapsed time or accumulating projected execution time. Library macros facilitate development of such specifications.

AUTO-G

Timing can be specified on events and processing paths in absolute or relative form.

DCDS

Performance criteria are attached to every function, to define probability of successful results. Low level timing can be specified on paths between validation points.

STP

No.

TEAMWORK

No. With the user definable dictionary template in Version 2.3, such information could be captured as an annotation in the database, and subsequently output in a report or interpolated into a process specification template.

3.3 Does the tool provide for alternative representations of the architecture elements (e.g., entity-relationship diagrams or Jackson structure charts in lieu of a textual data dictionary)?

TAGS

Only one view or method of representation is available for each part of TAGS.

AUTO-G

Only one method of representation is available in the G language -- trees representing the hierarchical decomposition. The textual T language may be used alternatively. Automated translation between those two languages is available at any time. ASA will soon permit T to be used on a G diagram as an alternative to the G graphics.

DCDS

The graphical F_net and R_net can be described in text as an alternative.

STP

Only DFDs are provided for functional system specification. Data flow and data store content can be defined using either Data Structure Diagrams or Entity Relationship Diagrams. Various diagrams and tables (e.g., state transition diagram, decision table, state event matrix, etc.) are available for definition of control aspects. STP does not provide any automatic translation from one to another. The user may set switches to view data flows, control flows, or both.

TEAMWORK

Teamwork provides for data or control flow diagrams or both, ER diagrams, structure charts, and state transition diagrams, process activation tables, or state-event matrices for control decomposition. Neither Teamwork nor other tools support the automatic generation of one formalism from another.

3.4 Does the tool's model of a process and mechanisms for process communication force upon the designer any design decisions? For example: Is it easier to design a centrally controlled system as opposed to a distributed control system, or *vice versa*?

TAGS

In general, no. However, TAGS supports a top-down decomposition approach to system design. The decomposition of an SBD component is viewed at the communication level; that is, the next-level down decomposition of a component is a view of the constituent components and their communication interfaces. The data that travels over each interface is well defined in a corresponding Input-Output Parameter Table (IOPT).

The TAGS model for communication is synchronous, but it is possible to model asynchronous communications.

AUTO-G

No.

DCDS

Yes, the tool forces the designer to make decisions about subsystems early in the design process. It also requires the designer to make order of execution decisions early in the design process since F_nets are time-dependent.

STP

It is easier to design centrally controlled systems. It is easier to proceed top-down in original composition, since input/outputs can be inherited by a child diagram.

TEAMWORK

It is easier to design centrally controlled systems. It is easier to proceed top-down in original composition, since input/outputs can be inherited by a child diagram.

3.5 Does the system provide for separate and largely independent specification of subsystems, processes, data flow, control, and data, to support alternative methods and patterns of architecture definition?

TAGS

In originally creating a design, a top-down approach, working from the SBD to the other tables and diagrams, seems preferable. With TAGS Configuration Management, a user may catalog alternative proposals or architectures as separate Engineering Change Proposals (ECPs). A "map" name is used to identify these alternative system constructs for engineering analysis. Rejected ECPs are retained as an historical record for life cycle maintenance. Approved ECPs become Specification Change Notices (SCN).

AUTO-G

AUTO-G is based on an integrated specification of all aspects of a design, and does not force a particular procedure or series of steps on the designer, except for a basic top-to-bottom progression.

DCDS

In practical terms, the required database translations force one to follow the recommended phases in the method. DCDS supports alternative hardware configurations within one database. The user does not have to choose between alternative hardware architectures until the MDL and TSL languages are used.

DCDS has no data flow diagramming capability. However, data flow and control flow can be recorded using the text editor.

STP

A top to bottom progression, with flow diagrams completed before the dictionary entries, and other tables and charts done last, is the typical pattern.

TEAMWORK

A top to bottom progression, with flow diagrams completed before the dictionary entries, and other tables and charts done last, is the typical pattern.

3.6 Describe any capability for incorporating and tracing original textual requirements.

TAGS

The configuration management utilities within TAGS provides a formal means for recording original textual requirements and a baseline system specification. Changes in specification would be handled by a formal change procedure. An audit trail of design changes can be obtained. A new package, RVTS, provides automated requirements tracing from requirements design specifications, however, this has no automatic interface to the current TAGS system.

AUTO-G

AUTO-G documents may be used to capture original requirements text and may be tied to the system design at the document levels. No automated mechanism exists for tracing associations.

DCDS

Throughout the system an item may be traced back to the originating document and various design decisions, since this information is stored in the database.

STP

Text annotations or comments on a diagram could be used to relate a design to textual requirements. No automated tracing mechanisms are provided.

TEAMWORK

No facility for relating functional requirements to a system specification exists other than to make text notes, which the system ignores. No facilities exist to facilitate the specification of hardware or network architectures or to specify timing constraints for a real-time system. In Version 2.3, Teamwork supports the inclusion of free form graphical notes.

3.7 How large a system could this tool accommodate, so far as you can tell?

TAGS

A present user, Unisys, states that they are now developing a naval navigation system consisting of more than 2000 pages and 100 documents (SBD components).

The performance and memory capacity of the host machine may impose limitations on the size of the system.

AUTO-G

IDA has not evaluated this as yet. The performance and memory capacity of the host machine may impose some limitations on the size of the system.

DCDS

The evaluators did not try to determine a limit on the database size.

STP

No pertinent limits are documented, and IDA did not try to determine any.

TEAMWORK

IDA has not evaluated this factor as yet. It depends on host performance and design methods used, among other considerations.

3.8 How many levels of decomposition does the tool support?

TAGS

TAGS supports seven levels of SBDs at this time.

AUTO-G

There are no published limits to the levels of decomposition that AUTO-G supports.

DCDS

The evaluators did not try to determine a limit to the number of levels of decomposition.

STP

A maximum of eleven levels of process decomposition is the limit on dataflow diagrams. More than eleven levels can be accommodated if the user turns off the option in IDEtool to show the full process index, since the limitation is on the size of the label in the process bubble.

TEAMWORK

The limitation on the level of process decomposition is only the name string for a process, which is a concatenation of all the parent process id numbers and the current process id number and is limited to 64 characters including the separators.

3.9 Does the tool actively assist the generation or development of software to be written in Ada? Does it produce SADMT automatically from a high level design? Does the tool allow representing tasks, packages, and generic units?

TAGS

The TAGS simulation compiler generates Ada as an intermediate representation. However, there is no means for a designer to control or identify tasks, packages, or generics. The Ada code generated by the SC is primarily for interpretive execution. However, the Ada Code Generator provides the user the ability to associate IORTDs with tasks, and parameter tables to packages. SADMT generation development is planned for 1988.

AUTO-G

G and T languages are very well suited to Ada semantics. ASA has a commitment to the Royal Military College to produce a G-to-Ada translator within the 1988 time period. In addition, ASA has delivered a G to SADMT translator for SDI applications, and is updating it for SADMT version 1.5.

DCDS

Ada code will be generated to support simulation. SADMT or any Ada PDL is not supported.

STP

Data types can be generated in Ada, Pascal or C. There is no automatic generation of procedural PDL, SADMT, or code text. The system has no simulation capability.

TEAMWORK

SADMT is not produced automatically, nor any procedural code text. Support for the design of Ada language applications is forthcoming in Version 3 and later releases. Version 3 will include an Ada design editor based on Buhr graphic notations. Also planned are a context specific window interface for text editing, and static checking and interpretive execution of Ada code.

3.10 How well does the native design language map to SDI's SADMT? For example: Which SADMT constructs does it provide/not provide?

TAGS

It is believed that mapping to SADMT would not be difficult. The block diagrams used to specify input/output procedures between elements (IORTDs) and procedures and macros (PPDs and MACs) seem to fit well with SADMT concepts.

However, the SADMT mechanisms for inter-process communication and IORL's mechanisms are radically different. SADMT uses an asynchronous message passing strategy and IORL uses synchronous sends and receives. It may be difficult to handle this mapping situation.

AUTO-G

Completely, at the present state of the SADMT simulation environment.

DCDS

Given the hierarchy of specification languages involved (SSL, RSL, MDL), it isn't clear how easily the SADMT process hierarchy could be generated. In principle, the same constructs exist as in SADMT.

STP

Only in respect to mapping the hierarchy of process bubbles to SADMT processes. It is not possible to represent packages, generics, hardware configurations (i.e., processors, memory and switches), timing constraints, details of process communication, etc., in the STP design languages.

TEAMWORK

The process hierarchy conforms to SADMT requirements, but no formal specification of leaf processes, timing requirements, or hardware allocations is provided.

3.11 Is the user provided with the option to specify process behavior by means of Ada text?

TAGS

No.

AUTO-G

No. T language text will be available soon as an alternative to G icons.

DCDS

No.

STP

Process behavior can be described by any text, including Ada text. This is transparent to the tool.

TEAMWORK

Teamwork provides an empty text specification template for each module. Teamwork does no processing of any text entered. Version 2.3 allows a user defined template. Ada language support is planned for Version 3 in 1988 and later.

4. DIAGNOSTICS

4.1 Does the tool provide syntactic and semantic checking between diagrams, data dictionary entries, and between data dictionary entries and diagrams?

TAGS

Syntactic and static semantic checking of the diagrams is done by the DA (diagnostic analyzer) and is presented through the diagnostic comments, dictionary, and cross reference tables (discussed above).

The checkout and validation of a design requires use of the DA. It may be applied either for the full system or for a part. Moreover, the analysis is apparently cumulative, in that parts of the analysis are retained, and a new version is analyzed only as far as its changes are concerned. Thus the first analysis of a new and large system is somewhat time-consuming, while the next analysis (with only a few changes or added block diagrams or tables) will be faster. It therefore would be possible to both make incremental analyses or go to a reanalysis regularly. Self checking for consistency during decomposition is NOT provided, but the analyzer may be run for the SBDs while they are first being defined.

AUTO-G

Because the G editor is syntax directed, any G-tree is guaranteed to be syntactically correct. Static semantics checking of the tree is done by the semantic analyzer, SEMA, and its results are presented to the user as diagram annotations, which is particularly convenient for debugging.

DCDS

The tool checks completeness and consistency between hierarchical levels, and checks for undefined entries in the data dictionary. Diagrams are checked for syntactic correctness before being saved, and cannot be saved in some instances unless correct. Users may add their own consistency and completeness checks through database query files.

STP

The tool provides syntactic checking between data flow diagrams, and between them and the data dictionary. Checking of control flows and their specification tables apparently was not fully implemented in the version provided to IDA.

TEAMWORK

Teamwork supports syntactic and semantic checking between diagrams of the same type in accordance with rules defined in the reference manuals. Both existence of data dictionary entries corresponding to flows or variables on a graphic diagram and the syntax of the entries may be checked. The user can choose to diagnose a given diagram or table, a subtree within a design proceeding down in decomposition from one selected level, or the entire system design. In Version 2.3, diagnostic checking can be done in background mode outside of Teamwork.

4.2 Are level-to-level associations maintained in the data dictionary? For example: When you explode a process into a subprocess, are the relationships between objects in the parent process maintained and enforced in the subprocess?

TAGS

The DA is excellent at enforcing that I/O interfaces be picked up at lower level SBDs and that IORTDs only reference interfaces attached to the corresponding SBD component.

AUTO-G

Yes, this is a key factor in the G language.

DCDS

The system checks if each data item is produced or consumed, but it does not automatically check if a function has been properly exploded. This check requires manual entry of each decomposed function and comments to check the interconnections. (See comments of the Phase 3 checkfile on pathflow and item flow).

The database is not completely integrated, although an integrated database is maintained on a per-language basis. A separate integrated database is maintained at each stage of the development. The pertinent information in each database can be downloaded to successive databases.

STP

Yes, when a process is exploded the external data flows are carried into the decomposition automatically. Parent and child diagrams are kept separately. Relationships are checked when a diagram's data dictionary is generated or the system decomposition is checked.

TEAMWORK

Level to level associations are maintained in the Teamwork dictionary. These associations are checked when the user explicitly invokes checking.

4.3 Do the diagnostic messages provide enough information to help correct errors?

TAGS

Error messages usually give a sufficient indication, but a document with further explanation would be helpful. Error messages are keyed to reference numbers automatically placed on diagrams by the system, so that the source of the error is easily located most of the time.

AUTO-G

Yes, in most cases.

DCDS

The graphical editor BROWSE merely states that there is an error and does not have diagnostic error messages. QUERY's checking identifies what type of error has been discovered. Messages are explained in the manual pertinent to the language involved.

STP

Yes, but problems do arise in the graphics where error sources become invisible and there is no way to correlate the messages in the error window with the diagram. An exception was found when an error in a data structure caused the unhelpful message "Error on line 12," even though there are no "lines" in these diagrams. If an anchor point is unconnected, a diagnostic message informs the user but does not highlight the point, which may be very difficult to find. It would be helpful if items causing an error were highlighted.

TEAMWORK

IDA had no problem understanding the diagnostic messages. The reference manual contains a further explanation of each error message.

5. DESIGN SUPPORT

5.1 Does the tool provide interfaces to other front-end design tools, data bases, project management software, etc.?

TAGS

No.

AUTO-G

No.

DCDS

No.

STP

Yes, it provides an interface to a version control system, SCCS, in the case of UNIX implementation. The underlying database package, TROLL, is accessible. There is also the tool library, a set of C functions that can be used by user-written programs. Integration with Sun's Network Software Environment has been announced for 1988 release.

TEAMWORK

Teamwork/DPI interfaces with Interleaf, Context , and Scribe publishing software. Teamwork also has been integrated with Sun's Network Software Environment for configuration management and source code control, and with VAX set CMS, a VAX configuration management system. Teamwork/ACCESS facilitates user effort to interface with desired support software.

5.2 Can data be imported/exported from the host data dictionary? If so, describe how.

TAGS

A system design can be put on media for backup or transport to another TAGS installation on identical hardware. Other import/export requirements would have to be provided via user prepared software.

AUTO-G

No importing is possible except as T language or user prepared software. Exporting as a text file to Unix is possible, but subsequent processing must be user provided.

DCDS

Yes, data can be imported and exported from the data dictionary. Exporting data is accomplished by creating an ASCII file from a QUERY. Importing is done using the @INPUT command, and requires the information to be in ENTRY's syntax.

STP

Yes, data import/export commands, provided in the main menu, upload or produce an ascii file.

TEAMWORK

Data can be imported/exported easily between versions of Teamwork running on different computers using Teamwork's dump/load capability. Using its ACCESS mechanism, users can write application programs which IMPORT/EXPORT data dictionary elements in any format. Cadre is currently working on a design database interchange standard based on a contemplated Electronic Data Interchange Format (EDIF) standard. This national standards activity appears to be gathering strong interest among tool producers and users.

5.3 Does the tool support macros or command files for reusing design elements or expediting frequent tasks?

TAGS

TAGS provides user-definable, parameterized macros for specifying I/O events in a macro library. A basic macro library is provided. In addition, PPDs (i.e., recursive logic) can be catalogued in PPD libraries for reusability. In 1988, a PPD index cataloguing tool will be available. This is based on a relational data base for subsequent ease in retrieving the indices via key word characteristics.

AUTO-G

Not as such, but the G template feature provides a powerful generic capability to support design reusability. The "document" organization is also helpful here.

DCDS

Yes. The INPUT feature is used for creating macros or command files.

STP

Design elements can be reused with the "Copy Diagrams" mechanisms and the cut/paste mechanisms. Many of the features of the Software Through Pictures environment are separately executed programs, which can be combined into shell scripts on a Unix system, and then added to the selections available from IDEtool by modifying the IDEtool specification file.

TEAMWORK

Teamwork has cut/paste and copy mechanisms to reuse design information.

5.4 Is there a subset of tools that can be used for designing a system's data base architecture? For example: Is there a tool for helping the user to normalize relational data base?

TAGS

No.

AUTO-G

No.

DCDS

The only tool for designing the database is the structure editor for I_nets. There are no tools to normalize the database.

STP

There are editors for creating Chen entity-relationship diagrams and Jackson data structure charts, but there are no tools to check if such a data base is in a normal form.

TEAMWORK

Teamwork provides support for the creation of entity-relationship diagrams. No database normalization tools are provided with the entity relationship diagram facility.

5.5 Does the tool accommodate the design of systems that incorporate artificial intelligence techniques (i.e., expert systems, natural language parsers, query-answer systems, etc.)?

TAGS

No specially tailored features exist, but its basic capabilities are applicable.

AUTO-G

No specially tailored features exist, but its basic capabilities are applicable.

DCDS

No specially tailored features exist, but its basic capabilities are applicable.

STP

The tool includes the RAPID/USE language, which can be compiled to recognize input strings and execute associated code.

TEAMWORK

No specially tailored features exist, but its basic capabilities are applicable.

5.6 Does the tool provide analysis techniques that identify the optimum design from a group of alternative designs? If so, how do these work?

TAGS

There are no tools that automatically identify optimal designs, but the TAGS Simulation Compiler, along with CM ECPs, allows the user to perform both static and dynamic analyses of alternative design concepts. Thus manual trade-off studies of critical requirements design data allow the engineer to select the optimal algorithm or system architecture.

AUTO-G

No.

DCDS

No, there are no analysis techniques in the tool for identifying optimum designs.

STP

No.

TEAMWORK

No. ACCESS supports user written analysis programs and metrics.

6. ADAPTABILITY

6.1 Does the system allow users to define their own structured method?

TAGS

No.

AUTO-G

It neither hinders nor explicitly supports a user community from following its own procedures for developing G specifications on a given project.

DCDS

Yes, the system allows the user to define their own method and to construct their own checks for their method. User-defined methods are supported for use with DCDS's text editor, not with the graphics editor.

STP

Not without extensive programming in C. The system utilities, PICture Editor, Tool Library, and Tool Info control file provide some basic capabilities for this purpose.

TEAMWORK

No. Teamwork's designers adopt the philosophy of protecting users from the inconsistencies they may introduce if allowed to arbitrarily adjust icons and design semantics. The Graphic Notes editor and open database architecture in Version 2.3 permit a user to augment Teamwork design methods, rather than modify them.

6.2 Does the tool come with an icon editor to permit user design of icons?

TAGS

No.

AUTO-G

No.

DCDS

The graphical editor cannot change an icon display or meaning. The syntax of the ENTRY system is fixed, but users can create their own entities and attributes.

STP

There is no icon editor. The picture editor provides a limited capability for modifying existing icons and creating new ones, but none of these modifications can be used in any other graphics editor.

TEAMWORK

No, but Teamwork version 2.3 provides an editor for user defined graphic icons as comments or annotations on diagrams.

6.3 Are user defined validation rules allowed?

TAGS

No.

AUTO-G

No.

DCDS

User defined rules are allowed and can be used in static checking.

STP

The user must supply his own code to implement any additional checking, but its activation can be coupled to the STP built-in diagnostics.

TEAMWORK

User defined rules must be programmed in C and invoked outside of Teamwork.

6.4 Does the system provide database access for linking user-written reporting or output software?

TAGS

The TAGS SC Manual shows examples of the output processing utilities and sample plots for accessing the raw SC data. Special license provisions can be obtained for read/write access to the TAGS data base (i.e., the IORL pictures). Documentation and read/write access library routines are then provided. This is being used by several TAGS users.

AUTO-G

ASA will provide database access information to users requiring it. The data dictionary text can be output and captured as a Unix file for user-controlled processing.

DCDS

The system has its own QUERY language, which would suffice for many kinds of querying. For other kinds of reports QUERY could produce a text file, which could then be used by a user-written program. There are no user-callable routines to access the database directly.

STP

Yes, through the Troll database utilities.

TEAMWORK

Yes, the Teamwork ACCESS facility provides this capability. Both read and write capability is provided in Version 2.3.

6.5 Does the system provide user access to the tool's architecture descriptive data base for adding user defined entities, attributes, and relationships serving such purposes as SADMT compatibility, requirements traceability, interface validation, etc.?

TAGS

If users write into the TAGS data base, they either must adhere to the DA syntax checking or they can use the comment symbol to enter special text with keywords. They could then scan the TAGS design data base searching for these keywords. Also, whole comment pages could be entered, as DA does not check comments.

AUTO-G

No.

DCDS

Yes, users can easily add their own entities, attributes and relationships using the Extend program, and can create files to check these user-written entries. This may be done at any time, even if data already exists in the database.

STP

The system provides a mechanism for including user-defined attributes and additional syntax (e.g. for process specifications).

TEAMWORK

With Version 2.3, users may write text entries into Teamwork's design database as user defined attributes. In Version 3, user-defined menus and forms will enhance the usability of these features for the purposes noted in the question.

7.0 PROJECT AND CONFIGURATION MANAGEMENT SUPPORT

7.1 Does the system provide version identification and time-date-stamping for different architectures derived from a common antecedent?

TAGS

Yes, version identification is available in the configuration management (CM) utilities, and time-date stamping is provided elsewhere, and used particularly by the diagnostic analyzer.

AUTO-G

Yes, every system or document (subsystem) may be marked by a user-chosen version label, and automatic version labels are added by AUTO-G as well. No time-date stamping is given, however.

DCDS

No, the system does not provide version identification or time-date stamping.

STP

No. However, it is expected to be used with the UNIX Source Code Control System for such support.

TEAMWORK

Version identifiers are automatically associated with diagrams and tables, 16 deep. Status labels on each diagram or table carry time-date stamping and such information as author, creation and modification dates, scheduled completion dates, etc.

A major limitation was that the versioning mechanism does not recall a consistent version of the entire design. The versioning mechanism only creates new versions of each object separately. The distinction is important since several object changes may be needed to effect one system change. To this end, Model Configuration Management capabilities will be included in release 3.0 of Teamwork.

7.2 Does the tool provide a project management utility? If so, describe its capabilities.

TAGS

No.

AUTO-G

No.

DCDS

No, there is no project management capability.

STP

No.

TEAMWORK

No project management support is provided directly. A sample reporting program is available under ACCESS that reads and lists status label information. This could serve as a basis for user-written report programs.

7.3 Can the project status be viewed for individual pieces of the system being developed?

TAGS

Yes, by using the Status Accounting Request menu (a CM option), a designer can obtain an up-to-date progress report for the entire system or a selected subsystem.

AUTO-G

No.

DCDS

No, the project status cannot be viewed for individual pieces of the system.

STP

No.

TEAMWORK

Yes, see 7.2 on previous page.

7.4 Do the project management facilities provide PERT/CPM or other types of analysis/planning tools for tracking the design effort?

TAGS

No.

AUTO-G

No.

DCDS

No, there is no project management facility.

STP

No.

TEAMWORK

No.

7.5 What type of configuration management capabilities does the tool have?

TAGS

TAGS provides a myriad of tools to support configuration management. Configuration Manager has built-in controls. All diagrams are date-stamped, but the overall system may also be placed under configuration management; though we did not try it, Teledyne Brown offered to show us how, but warned us it imposed constraints on updating (because the CM controls must be satisfied before change is allowed.) TAGS' CM supports the use of Trouble Reports, Engineering Change Proposals, Specification Change Notices, and Distribution Cover Sheets. These "forms" allow the precise definition of the system requirements and problems, possible solutions, well defined proposed changes, and a record of actual modifications.

AUTO-G

None. At least one user has coupled AUTO-G to its proprietary CM software.

DCDS

None.

STP

None. IDE has announced coupling to external CM package for 1988.

TEAMWORK

Teamwork has been coupled to other configuration management software, see question 5.1; the Model Configuration Management package is expected in 1988 in Version 3.

7.6 Does the tool support traceability for decisions and changes over the entire system life-cycle (e.g., baselining and versioning)?

TAGS

Yes, Trouble Reports, Engineering Change Notices, and Specification Change Notices exist to make decisions and changes, once a design has been placed under CM. The Status Accounting Request menu, the Update Request menu, and the Audit Request menu give the user sufficient means to keep track of what has been proposed, approved, and implemented.

AUTO-G

No.

DCDS

No.

STP

No.

TEAMWORK

Teamwork provides some object modification traceability through use of the Status Label which can be filled in on each version of an object.

7.7 What progress metric, if any, does the system produce? Is it a quantified measure of completeness?

TAGS

The TAGS Audit tool provides a count of design pages for each system component in support of AFSCP 800-43 and 800-14. The system is scheduled to generate metrics in early 1988 in CM on trouble reports, their frequency, specific modules, magnitude of ECPs, time duration of trouble report, etc.

AUTO-G

None.

DCDS

The system is used in a fixed sequence of phases, and the number indicates how complete the system is. It is not a reliable guide, since later stages may demonstrate an error and force the designer to go back a few stages.

STP

None.

TEAMWORK

Version 2.3 incorporates a menu selection to compute the Bang complexity rating [DeMarco].

7.8 Can parts of the architecture definition be marked as draft, under review, approved, etc.?

TAGS

When placed under CM, using the Engineering Change Proposal form, any portion of a system may be marked according to such status categories.

AUTO-G

Yes, by user-authored version labels.

DCDS

DCDS provides for tagging of architecture sections' approval rating via the COMPLETENESS attribute.

STP

No.

TEAMWORK

Yes, status labels serve to do this.

7.9 Does the tool support integrated work by multiple analysts/workstations (e.g., a local area network)?

TAGS

Yes.

AUTO-G

Yes, a central database on a server is workable by using Unix protection, working copies, and having a project administrator.

DCDS

DCDS does not support more than one analyst working on the same database.

STP

Yes. The file locking system must be enabled for this. Software Through Pictures also supports heterogeneous networking through the use of the Network File System and/or XWindows so that users can work on different kinds of machines, including non-workstation servers.

TEAMWORK

Teamwork provides multiuser concurrency control and distributed processing capabilities (see network capabilities listed in Appendix A). These are transparent to users. Teamwork's distribution is as follows: a central project database server resides on a single node, having multiple models, and communicates with remote nodes running the Teamwork graphics processes.

8. SIMULATION CAPABILITIES

8.1 Describe the tool's dynamic tests or simulation on an architecture definition.

TAGS

First, the DA checks the syntax and static semantics of the a system as defined in IORL. Once this has been satisfactorily completed, the dynamic semantics are analyzed by the TAGS simulation environment. The SC generates two files: an ASCII trace listing and a binary trace file. Users then invoke the SC access utilities to process the binary file to generate reports, graphs, and tables on their particular installation's output devices.

The dynamic semantics are analyzed by the TAGS simulation environment, which consists of a simulation compiler (SC) that produces an executable model based on Ada. Further static errors may be located during the SC phase. Finally, an Executable Code Generator (ECG) produces Ada code that is passed via a network or magnetic tape to a VAX (using the VMS operating system) for computation and simulation.

Simulation may be carried out at different (user defined) levels of detail. This is achieved by selecting the SBDs and IORTDs of interest at the level being simulated. This is performed during the first phase of the SC, thus producing a "blueprint", which is separately named and may be retained for further work, if needed. VERIFY and BUILD processes then check for further errors and produce the model, then export the Ada code.

Compilation of the Ada code currently uses the DEC Ada compiler, though there may be working compilers at the workstations in the future. During simulation, three further inputs are needed: types of trace file messages needed; error message limits for aborting the simulation; and the run-time parameters. These may be specified during the "simulation run setup session." Input to the simulation may be either in files or the user terminal. Results are presented in reports, graphs, or tables.

AUTO-G

Currently, no simulation is provided but development is underway on one called AUTO-X.

DCDS

There is no dynamic checking capability at this time, but a simulation capability, from generated Ada code, is available as of November 1987.

STP

None.

TEAMWORK

None at present. Cadre is developing a token-based simulation for Version 3, which will allow dynamic checking of flow and activation.

8.2 Can you simulate just a subset of a system and/or use different levels of detail?

TAGS

Yes, TAGS allows the user to specify exactly what the user wishes to simulate. That is to say, if the designer has an IORTD associated with a component which is further decomposed, he or she may choose to simulate the system with the behavior for the component as specified in its IORTD or choose to simulate the system by using the behavior of the constituent component as specified in their IORTD's.

AUTO-G

Unknown.

DCDS

Unknown.

STP

N/A

TEAMWORK

N/A

9. DOCUMENT PRODUCTION

9.1 What facilities does the package support to print its diagrams and tables?

TAGS

Primary hardcopy output is on POSTCRIPT based printers, such as the Apple Laserwriters. A dot matrix printer capability also exists.

AUTO-G

The Hewlett Packard 7475 plotter and compatibles are supported.

DCDS

Both the AT version and the VAX version supports Postscript formatted output files (not tested by IDA).

STP

It supports Postscript, Unix pic, or raster formatted output files and laser printers or plotters that can print them.

TEAMWORK

Teamwork generates documents in Interleaf, Context, or Scribe formats.

9.2 What is the maximum amount of textual description allowed per object on graphic diagrams? Can it be included on diagram prints?

TAGS

Besides the text utilities provided in CM, the graphics editor allows the user to make free form comments, without any real limit other than screen size. For example, PPD comment pages could include the SADMT syntax or Ada PDL.

AUTO-G

Any rectangle can be filled in with as much as the user desires.

DCDS

The evaluators were unable to find a limit to the different types of textual descriptions per object, as they can be user-defined, and there is no apparent limit to the amount of text per entry.

STP

The system allows a user to use 480 character lines to label a diagram element. All of this is stored in the diagram and the data dictionary, but only 12 characters are printed and, in the case of diagram name, only 20 are treated as significant. No limit on data and process specification text was encountered. The only provided attributes for data elements are type, constraints, aliases and selector. There are no attributes for process specifications, although the system does list the input and output data flows and provide a space for a textual description. This description is not subject to any form of checking.

TEAMWORK

Teamwork provides for 64,000 bytes of textual description per object. In version 2.3 of Teamwork, object types may have user-defined attributes incorporated into their data dictionary entries.

9.3 Does the system provide automated options or a general report/graphics generation capability for documentation and graphics derivable from its dictionary, such as printed or plotted diagrams, hierarchical indices, interface control tables, parts lists, cross reference charts, and other standard reports on an architecture?

TAGS

A dictionary listing, cross-reference reports, decomposition trace, data flow trace, index of all documents, date and time change reports, and other standard reports are available. Selected dictionary items may be queried.

AUTO-G

AUTO-G provides no separate reports or output other than the graphic design created by the user or the error annotations to it produced by SEMA. A data dictionary listing may be viewed in a separate listing and captured in a Unix file for separate processing. AUTO-G provides a way to search diagrams for a given name.

DCDS

The system provides a general report generation capability through QUERY, which can produce hierarchical indexes, interface control tables, and other types of reports.

STP

The system provides the capability to output (print) either a portion or the entire data dictionary through the Data Dictionary Program. In addition, the Data Dictionary Program can also automatically create and print a table of contents for the data dictionary. Dictionary can be queried selectively through the dictionary browse feature.

TEAMWORK

Teamwork provides a document preparation interface (DPI) which, with Teamwork's ACCESS, can be used to produce dictionary reports. The user must, however, program or edit each report. No standard reports are automatically generated from the data dictionary.

9.4 Does the system support DoD standard documentation formats?

TAGS

The system interfaces with the CONTEXT Corporation document processor. CONTEXT supports the cataloguing of DoD-STD-2167 templates. IORL pages are converted to POSTSCRIPT, and CONTEXT then allows automated cut-and-paste and scaling and rotation of the IORL pages into DoD-STD-2167 documents.

AUTO-G

DoD standard documentation is not supported.

DCDS

DCDS supports Mil. Std. 2167 (DoD standard documentation).

STP

No DoD standard formats are supported. IDE has announced interfaces to the technical publishing systems of Interleaf, Frame Technology, and Scribe Systems, as well as support for the MIL-STD-2167 Data Item Definitions. One of IDE's users has implemented a preliminary version of this capability by extensions to Software Through Pictures. This product will not be available until early 1988.

TEAMWORK

Yes, Teamwork/DPI provides templates for all Mil-Std 2167 DID formats which greatly facilitate the creation of these documents.

9.5 Does the tool support the creation and cataloging of informal graphics and tables to illustrate aspects of the designed system?

TAGS

TAGS comment pages allow free form text. The Data Structure Diagrams (DSDs) allow forms and table creation.

AUTO-G

No.

DCDS

No.

STP

It supports creation of informal graphics and tables, but does not catalog them or tie them into the system specification.

TEAMWORK

Graphical and textual annotations are catalogued under Version 2.3.

9.6 Is the documentation of the tool adequate and readable in describing the tool's capabilities and operational use?

TAGS

Documentation is extensive and does not appear to be worse than typical for a package of this size. The Input-Output Requirements Language (IORL) is well documented. The IORL manual provides a full specification. We did not rely on documentation to answer most questions, but depended instead on direct contact with development personnel.

However, the IORL as defined in the language reference manual and the IORL that is acceptable to the DIAGNOSTIC ANALYZER are different. Tool implementations are limited by machine and technology constraints. Therefore the tools provide a practical and working subset implementation of the language. The limitations are noted in each manual. Producing a specification consistent with the IORL defined in the language reference manual may result in errors by the DA. An example of this is in the use of library macros for communication, which one said was available but which are not. Another example is the manual's allowance of global variables (at the IPT-O level) whereas only constants will pass the DA.

AUTO-G

The documentation provided is complete but not indexed, hence it may take time to find specific information. The G and T reference manuals are well written. The Toolset Users' Guide lacks thorough descriptions and command examples, but does provide a description of error messages. Several complete design examples for practical systems are available, a distinct advantage.

DCDS

The documentation lacks a master index, so finding information was often difficult. The documentation is generally complete, but large sections are out-of-date, documenting the older Pascal version instead. The documentation covers the language, method, and tools, though the language presentation is tediously long and not usage oriented. The documentation also covers the tools and their interrelationships, though the actual commands to execute certain steps are vague occasionally. The diagnostics do not highlight errors well but are explicit in the kind of error found.

STP

The documentation on the tools, diagnostic messages, and the Troll database is complete. However, there is no documentation available on the methods supported by the tool. A table of contents is provided for each section of the User Reference Manual, but no overall index.

TEAMWORK

The feature documentation is well-indexed, complete and current. Error diagnostics are well documented. No documentation is provided for the methodologies. The interrelationship of the various Teamwork tools is minimally documented.

10. HARDWARE CONSIDERATIONS

10.1 From an ergonomic perspective, how would you rate the tool? For example: Were you able to adjust the size of the text or objects on the screen? Are the keyboard and mouse well designed? Was it quite comfortable to use the tool?

TAGS

Very good.

AUTO-G

Very good.

DCDS

The man-machine interface is poorly designed. DCDS supports the Tektronics, VT240, and PC monitors. The resolution of the PC monitors is usually 640 x 350 pixels. The resolution for the Tektronics monitor is 1024 x 768 pixels. The VT240 only supports the use of DCDS in text mode. In the AT version, the mouse cursor moved to the center of the screen after every operation. The size of text or objects could only be adjusted for all objects, not for a subset of object types.

STP

The ergonomic design of the tool is very good. The tool supports high resolution graphics, scaling, multiple windows, and movement through the windows via an optical mouse.

TEAMWORK

IDA users were very comfortable with Teamwork on both Apollo and Sun workstations.

10.2 Is the tool ported to other hardware configurations?

TAGS

TBE is porting the system to X Windows Version 11 and Unix V.3 interface standards.

AUTO-G

Currently, it is hosted on the Sun, Apollo, DEC Vax Station, and Atari 1040ST, and may be used on any Unix 5 system with a graphics terminal, with some performance loss.

DCDS

No.

STP

Software Through Pictures is available for all models of the Sun Workstation; the DN 580, 3000, and 4000 series of the Apollo Domain workstation; the HP 9000 series 300 and series 800; the VAXstation II, 2000, and 3000 series under both Ultrix and VMS; and the Sony NeWS workstation. A MASSCOMP implementation is underway.

TEAMWORK

Apollo, DEC VAX station, Sun, HP 900-300, and IBM PC-RT. Teamwork's database can be hosted on any VAX machine.

Appendix C

Design Exercise -- Elevator Control

Objective and Scope

This exercise is part of an evaluation effort to determine the useability and effectiveness of certain computer-aided design systems. Using the designated computer-aided design system, design a microcomputer software system to control and direct the operation of M elevators in an N floor high rise office building. The control system should provide fully automatic operation of all elevators, and accept the human input necessary to conventional elevator service, such as requesting to board an elevator going up. In addition, for monitoring and direction, there is a supervisory information display and command terminal. Specific requirements and directions for conducting the exercise are given in the following text.

Criteria for a Design and Conducting the Exercise

This design exercise is set up in two stages. The requirements as outlined below are for the first stage, which describes a control system where each elevator is hardwired to the master controller. The second stage, outlined later, provides for distributed control of the elevator system (i.e., not hardwired).

In the initial attempt at completing the first stage of the exercise, the onboard computer, elevator movement, door operation, onboard displays, etc. do not have to be decomposed or designed. Rather they are represented as a single external input-output interface to the master control subsystem. This interface transmits the elevator's status message to the master and receives the commands from the master at the appropriate time intervals. At some level of decomposition, the commmunications protocol at the interface will have to become visible in the specification. (This approach conforms to the simulation concept for SDI Battle Management architecture. Nevertheless, recognizing that the responses of this interface are correlated over time with the input from the master, and vice-versa, a valid simulation will have to include appropriate intelligence that produces such responses at the interface.) Note however that the on-floor buttons and lights, and the

supervisor command or query interfaces to the master must be defined. Also, failure behavior for the master must be modeled.

Control Systems and Control Regimes

There are two alternative control systems: a hardwired, centralized control system (stage 1); and, a distributed control system (stage 2). Each control system provides two alternative control regimes that represent alternative approaches to achieving the performance goal of minimum average time between a passenger boarding request and the arrival of the passenger at the requested floor. "Control regime" refers to the overall approach to coordinated control of all elevators, including the allocation of each elevator to a specific type of service, determination of the elevator responding to a pending request, and the control of elevator state changes (stopping and reversal). The possible types of service for an elevator are:

* out-of-service: stopped indefinitely at some floor;

* normal: serving all floors and passenger requests, one direction at a time;

* express: not serving (skipping) a contiguous set of floors.

Both control regimes must be "passenger friendly", meaning that any elevator continues in one direction until all on-board passengers proceeding in that direction have reached their floor.

A given control regime provides a certain resolution of questions such as: when an elevator becomes empty of passengers, shall it continue in the direction it was taking, to answer other boarding requests? would it be advantageous to have empty elevators proceed to predetermined positions to await future boarding requests? is it advantageous to have a mix of prepositioned and circulating elevators?

Display and terminal capabilities

Human command and control of elevator activity is essential, and is provided through a conventional keyboard/display terminal, placed along with the master microcomputer and associated equipment in some fixed location in the building. By suitable key actions, the human supervisor may select one of the two alternative regimes, or may input parameters or constraints for a given control regime, such as placing an elevator out of service or designating a specific elevator to provide express service. In addition, direct manual action (requiring a conventional key to operate a switch) shall be possible at any floor, to accomodate service and safety requirements. Please note that the human supervisor and the master controller are two separate entities.

At each floor, up request and down request buttons are provided for passengers (with obvious exceptions for top and bottom floors) to signal for an elevator. There are two types of buttons, to request either an express or a normal service elevator. If no express elevator service is available, or a person requests express service at a floor where express elevators will not stop, the display light will NOT go on, to indicate this fact. (As stated later, at least one normal service elevator shall always exist.)

A bell, a direction indicator light, and a normal vs express service indicator light above each elevator's door will be activated to announce an elevator's arrival at each floor (just before it arrives) and its status (current direction, express or normal service). A pushed request button remains lighted until a responding elevator arrives; also the elevator indicator lights on the floor remain on until the elevator closes doors and departs.

On each elevator, conventional displays and button controls for passenger safety and convenience are provided. For instance, a passenger may hold the doors and keep the elevator from proceeding by constantly depressing a "doors open" button. Each on-board passenger's stop floor request is shown via a lighted button.

First Stage

The basic control decisions affecting each elevator's service and their coordinated activity are invested in a master microcomputer, which also handles the supervisor interface, information processing, and command interpretation. The master computer communicates with separate microcomputers aboard each elevator. The onboard computer handles the primitive control functions directly associated with an elevator's displays and buttons, door control, startup, speed control, etc. The master computer controls the on-floor lights and announcement bell. The master computer samples each elevator's status (present floor, direction, and pending passenger departure requests) during each elevator stop. The master computer transmits commands, for example to set an elevator's next stop and to start it up, before the elevator proceeds from its present stop. If a failure occurs, certain default behavior ensues. If the master computer does not receive a status message from an elevator within a time-out period (may vary with the number of floors the elevator had to travel to next stop), it will signal the supervisor with an alarm message and then consider the elevator as out of service. If the onboard computer of an elevator receives no response from the master to its status message at its current stop, then after a time-out period it will begin executing a default behavior. The default behavior (Sabbath mode) is to proceed in its current direction, stopping at each subsequent floor, and reversing automatically when it reaches the top or bottom floor.

In other words, it will circulate, stopping at every floor. The following control regime is to be provided as a minimum. As stated later, a second regime is to be conceived and added for the exercise. Elevators operate only in normal service, as defined above, or are out of service. The master computer determines the movement and positioning of all elevators which are in service. At each elevator's stop, the master receives a status message, as above, specifying the elevator's current floor, last direction of movement, and all pending passenger requested floor stops. The master transmits back the next direction and next stop of the elevator, and a start-up signal that activates the primitive onboard control to close doors and move on. The master selects the next stop so as to maintain "passenger friendly" behavior (one direction until empty) and to minimize the waiting time of prospective boarders. An empty elevator may be reversed at any floor, and when no boarding requests are pending, empty elevators are placed out of service at their current floor. An out of service elevator may be reactivated by the master at any time, by sending a

"open doors" command (when needed at its present floor) or a "next stop" command and start-up signal to it.

The second control regime (designer's choice) is to provide express service (designated by the supervisor by one or more elevator numbers) and other control rules as designer decides might improve average waiting time. As appropriate, empty elevators also may be prepositioned at different floors, possibly depending on time of day, such as the first floor or some higher one, to await boarding requests. The supervisor may designate (as a parameter) a given floor for prepositioning of each elevator. Also add a decomposition and detailing of the onboard microcomputer software as an integral part of the overall system, rather than an external interface outside the system.

Second Stage

Now model the elevator system using a distributed control approach, as outlined below. At the top-most level, the elevator system can be viewed as containing four entities: the set of elevators, a controller-- which is a particular elevator that has control at a particular point in time, the environment, and the human supervisor. A fault introduced from an internal or external source could disrupt the functioning of one or more of these entities. The elevator system should be modelled such that when an elevator goes out of service, the other elevators and the controller can detect this event, modify the schedule to accomodate the new configuration of elevators, and get any passengers on the out-of-service elevator to their destination. Any elevator will be able to take over as the controller (i.e., each elevator's microprocessor can provide scheduling, etc.). A controller is only replaced by another elevator if the controller does not respond to a transmission from an elevator within a specific period of time.

Communications Protocol. A specific communications protocol for the second stage is described below to provide input for using the simulation capabilities of the tool you are evaluating. The communications protocol consists of the bus topology with the CSMA/CD (carrier sense, multiple access, with collision detection) control scheme. Broadband signaling using frequency division multiplexing (FDM) will be used. The bus provides a bidirectional transmission facility to which all of the nodes (i.e., microprocessors on the elevators) are attached. Information signals propagate away from the originating node in both directions to the terminated ends of the bus. Each node is

tapped into the bus and copies the message as it passes that point in the cable. The CSMA/CD control scheme permits any node to begin transmitting data whenever it detects that the bus is idle. The node continues to monitor the bus for interference from another node that may have begun transmitting about the same time. Any collisions will be detected by all transmitting nodes, causing those nodes to halt transmission for a short random time period before attempting to transmit again.

Broadband signaling employs analog signals and multiplexing techniques on the LAN medium to permit more than one node to transmit at a time. Frequency division multiplexing creates multiple channels (frequency bands). Radio-frequency modulator/demodulators (rf modems) are required at the sender and receiver. Use the following specifications for the system: Bandwidth: 300 MHz, Channel size: 6-MHz, Data rate per channel: 5 Mb/s. Note: Two adjacent 6-MHz channels can be used to provide a single 12-MHz channel for data rates up to 10 Mb/s.

Frames. Messages are transmitted via frames across the bus. Each frame consists of the following fields: Information field: the actual message; Delimiter field: marks the beginning and end of the message; Control field: contains status info., frame length, etc.; Address field: source and destination indicator.

The size of each field is a follows: Information field: 5 bytes; Delimiter field: 2 bytes; Control field: 2 bytes; Address field: 4 bytes.

Each elevator has a unique numeric identification. The identification number also indicates the type of service the elevator provides--express or normal. The information field contains the request or command that is to be sent across the bus. For example, an elevator could send a message to the controller that it received the controller's command to respond to a boarding request at floor number five, and is presently moving past the third floor at a velocity of 5 meters per second.

Fault Generation. The generation of faults is to be done using the simulation capabilities of the CASE tool you are using. Try to model only those faults that would cause an elevator or controller to go out of service. For example, the loss of the overhead light in the elevator would not be considered to be an important fault. Use milliseconds to describe transmission and delay time, and seconds as the unit of measure for simulating

out-of-service time. Use meters as the measure of distance travel, and meters/second as the measure of speed (could make the problem more realistic by including velocity and acceleration).

APPENDIX D

GLOSSARY

GLOSSARY

AEGIS:	A multiuser operating system for the Apollo Domain workstation.
Alpha:	DCDS' small-grained functions.
ASA:	Advanced System Architectures, vendor of Auto-G.
Auto-G:	ASA's graphical design language.
Auto-T:	ASA's textual design language.
BM/C^3:	Battle Management/Command, Control, and Communications.
Browse:	DCDS's editor.
Cadre:	Vendor of Teamwork.
CASE:	Computer-Aided Software Engineering.
CM:	Configuration Management.
Context:	A desktop publishing software package.
CSE:	STP's Control Structure Editor.
CSMA/CD:	Carrier-Sense Multiple-Access with Collision Detection.
DA:	TAGS' Diagnostic Analyzer.
DBMS:	Data Base Management System.
DCDS:	TRW's Distributed Computing Design System.
DDL:	DCDS' Distributed Design Language; defines architecture of H/W and S/W.
DDM:	DCDS' Distributed Design Method.
DFD:	Data flow diagram.
DFE:	STP's data flow diagram editor.
Domain:	Workstation made by Apollo.
DSD:	Data structure diagram.
DSE:	STP's data structure editor.

ECP:	TAGS' Engineering Change Proposal utility.
ERA:	Entity-Relationship-Attribute diagram.
ERD:	Entity relationship diagram, synonym for ERA.
ERE:	STP's entity relationship editor.
FDM:	Frequency division multiplexing.
F_net:	DCDS' Functional network; used to define a functional model of the system.
Graphic Notes:	An editor in Teamwork that provides object annotations in free form graphics.
H/W:	Hardware.
IBM RT PC:	IBM's RISC machine.
IBM AT:	IBM microcomputer based on the Intel 80286.
IBM PC:	IBM microcomputer based on the Intel 8088.
IDE:	Interactive Development Environments, vendor of STP.
IDEtool:	STP's main menu.
I_net:	DCDS' Item network; used to represent the time sequence and arrangement of data (items).
Interleaf:	A desktop publishing software package.
I/O:	Input/Output.
IOPT:	TAGS' I/O Parameter Table; specifies communication parameters between system components.
IORL:	TAGS' I/O Requirements Language; the graphical system design language, which allows the user to draw block diagrams of the system to be a specified and then document the modules and links to other parts of the system.
IORTD:	I/O Requirements and Timing Diagram; used to specify the algorithmic logic that corresponds to an individual component.
Macro:	TAGS' procedure call for an I/O operation.

MDL:	DCDS' Module Development Language; defines units of program code.
MDM:	DCDS' Module Development Method.
MERGE LIBRARY:	A TAGS tool which performs the merging of different system components from other geographical locations into one database.
MIL-STD 2167:	Military standard for documenting software.
PCT:	STP's PICture editor; a general-purpose graphics editor.
PIC:	STP's PICture editor; synonym for PCT.
PPD:	TAGS' Predefined Process Diagram; a procedure definition.
PDL:	Process Description Language.
QUERY:	DCDS' database query language; used to create reports and extract information.
RSL:	DCDS' Requirements Specification Language.
RVTS:	Teledyne Brown's package for tracking system requirements.
R_net:	DCDS' Requirements network; a directed graph defining the flow of control between alphas and specifies the system response to events and input messages.
SA/PDL:	SDI Architecture Process Description Language. Has been superceded by SADMT.
SADMT:	SDI Architecture Dataflow Modeling Technique.
SBD:	TAGS' Schematic Block Diagram; represent components of a system.
SC:	Structure Chart.
SCE:	STP's Structure Chart Editor.
Scribe:	A desktop publishing software package.
SEMA:	Auto-G's SEMantic Analyzer.
SREM:	DCDS' System REquirements Method.

SSL:	DCDS' System Engineering Language; defines systems and their function.
STD:	State Transition Diagram.
STE:	STP's State Transition Editor.
STP:	IDE's Software through Pictures.
S/W:	Software.
SYSREM:	DCDS' System Requirements Engineering Method.
TAGS:	Teledyne Brown's Technology for the Automated Generation of Systems.
Tasks:	DCDS' term for a cluster of R_nets.
TDE:	STP's Transition Diagram Editor.
Teamwork:	Cadre's Teamwork.
Teamwork/ACCESS:	Teamwork utility to access its data dictionary.
Teamwork/DPI:	Teamwork's Document Production Interface.
Teamwork/IM:	Teamwork's Information Modeling utility; used to implement Chen Entity-Relation diagrams.
Teamwork/RT:	Teamwork's tool for Structured Analysis with Real Time Extensions.
Teamwork/SA:	Teamwork's tool for Structured Analysis.
Teamwork/SD:	Teamwork's tool for Structured Design.
Teledyne Brown:	Vendor of TAGS.
ToolInfo:	STP's specification file for customizing STP.
Troll:	Database package used internally by STP.
TRW:	Vendor of DCDS.
TSL:	DCDS' Test Support Language.
TSM:	DCDS' Test Support Method.
Unix:	A multiuser operating system from AT&T.

VAX:	Digital Equipment Corporation computer model.
Visible Connections:	STP's set of published file formats which allow the user to customize the tool set.
VMS:	A multiuser operating system for VAX computers.
XQT Estimate:	DCDS' construct for specifying timing requirements for individual lines of code.

Distribution List for IDA Paper P-2062

NAME AND ADDRESS	NUMBER OF COPIES
Sponsor	
Lt. Col. Jon Rindt SDIO/SYS, Room 1E149 Pentagon Washington, D.C. 20301-7100	3 copies
Other	
Geoffrey Baum Martin Marietta P.O. Box 1260 Denver, CO 80201-1260	1 copy
Ernst M. Binder General Electric Co. Bldg. 8, Room 8722 P.O. Box 8555 Philadelphia, PA 19101	1 copy
R.T. Broacha General Research Corp. 5383 Hollister P.O. Box 6770 Santa Barbara, CA 93111	1 copy
Alice Brown Mail Stop 23 ECI Division E-Systems, Inc. P.O. Box 12248 St. Petersburg, FL 33733-2248	1 copy
Mr. Roger Carson Riverside Research Institute 1815 North Fort Myer Drive Suite 100 Arlington, VA 22209	1 copy
Subhash Chadha Navistar International Transportation Corp. 1901 S. Meyers Road Oak Brook Terrace, IL 60148	1 copy

Louis Chmura
Code 5533
Naval Research Lab
Washington, DC 20375-5000
1 copy

Judy Clapp
Mitre Corp.
Burlington Road
Bedford, MA 01730
1 copy

Carol Combs
National Security Agency
9800 Savage Road
Ft. Meade, MD 20755-6000
1 copy

Lee Cooper
Advanced Technology
2121 Crystal Drive, Suite 200
Arlington, VA 22202
1 copy

Larry Cox
TRW
1950 Sunwest Lane, Suite 302
San Bernardino, CA 92408
1 copy

David M. Davis
Systems Engineering & Analysis
The MITRE Corporation
7525 Colshire Drive
McLean, VA 22102-3481
1 copy

Ford Aerospace & Communications Corp.
ATTN: Jim Egolf (MS 38A)
10440 State Highway 83
Colorado Springs, CO 80921
1 copy

Mr. Everhart
Colsa Inc.
6724 Oddessy Dr.
Huntsville, AL 35806
1 copy

Agnes Fong
TRW R2/2020
One Space Park
Redondo Beach, CA 90278
1 copy

George Gearn — 1 copy
Applied Research & Engineering, Inc.
7 Railroad Avenue
Suite F
Bedford, MA 01750

David C. Hartmann — 1 copy
Washington C3I Networking Center
The MITRE Corporation
7525 Colshire Drive
McLean, VA 22102-3481

Norman Heck — 1 copy
IBM
1300 N. 17th St., Suite 7100
Arlington, VA 22209

John Hendricks — 1 copy
Systems Technology Inc.
242 Ocean Drive West
Stamford, CT 06902

Maris Juberts — 1 copy
Metrology B124
NBS
Gaithersburg, MD 20899

Joe Kadera — 2 copies
Satellite & Space Electronics Division
Rockwell International Corp. (SX-25)
P.O. Box 3644
Seal Beach, CA 90740

Vicki Kitchen — 1 copy
Applied Research Inc.
P.O. Box 11220
Huntsville, AL 35814-1220

Steve Nies — 1 copy
SPS
P.O. Box 361697
Indiatlantic, FL 32909

General Dynamics Space Systems Division — 1 copy
P.O. Box 85990
San Diego, CA 92138
ATTN: Cindea Metzler (MZ C1-8570)

Erich Muller
Sparta Inc.
7926 Jones Branch Dr.
McLean, VA 22102

1 copy

T.F. "Skip" Saunders
MITRE Corp.
Burlington Road
Burlington, MA 01730

1 copy

Dr. John J. Shaw
Alphatech, Inc.
2 Burlington Executive Center
111 Middlesex Turnpike
Burlington, MA 01803

1 copy

Heather Shuck
TRW
1 Federal Systems Park
Fairfax, VA 22033

1 copy

Steven Stendahl
WJSA
1901 N. Ft. Myer Dr., Suite 800
Arlington, VA 22209

1 copy

Martin Zlotnick
NRC
8618 Westwood Center Dr.
Vienna, VA 22180

1 copy

Defense Technical Information Center
Cameron Station
Alexandria, VA 22314

2 copies

IIT Research Institute
4550 Forbes Blvd., Suite 300
Lanham, MD 20706

1 copy

Vendor Contacts

Mr. Miguel Carrio — 1 copy
Teledyne Brown Engineering
3700 Pender Drive
Suite 200
Fairfax, VA 22020

Ms. Yvonne Cekel — 1 copy
Marketing Manager
Cadre Technologies, Inc.
222 Richmond Street
Providence, RI 02903

Mr. Larry Christina, Jr. — 1 copy
Technology Branch, CSSD-H-SBY
Battle Management Division
U.S. Army Strategic Defense Command-Huntsville
Post Office Box 1500
Huntsville, AL 35807-3801

Mr. Mike Guillebeau — 1 copy
TRW
213 Wynn Drive
Huntsville, AL 35805

Dr. Anthony Wasserman, President — 1 copy
Interactive Development Environments, Inc.
150 Fourth Street
Suite 210
San Francisco, CA 94103

Mr. Christopher Williams — 1 copy
Advanced System Architectures
Johnson House
73-79 Park Street
GU 15 3PE, United Kingdom

CSED Review Panel

Dr. Dan Alpert, Director
Center for Advanced Study
University of Illinois
912 W. Illinois Street
Urbana, IL 61801

1 copy

Dr. Barry W. Boehm
TkW Defense Systems Group
MS 2-2304
One Space Park
Redondo Beach, CA 90278

1 copy

Dr. Ruth Davis
The Pymatuning Group, Inc.
2000 N. 15th Street, Suite 707
Arlington, VA 22201

1 copy

Dr. Larry E. Druffel
Software Engineering Institute
Shadyside Place
480 South Aiken Ave.
Pittsburgh, PA 15231

1 copy

Dr. C.E. Hutchinson, Dean
Thayer School of Engineering
Dartmouth College
Hanover, NH 03755

1 copy

Mr. A.J. Jordano
Manager, Systems & Software
Engineering Headquarters
Federal Systems Division
6600 Rockledge Dr.
Bethesda, MD 20817

1 copy

Mr. Robert K. Lehto
Mainstay
302 Mill St.
Occoquan, VA 22125

1 copy

Mr. Oliver Selfridge
45 Percy Road
Lexington, MA 02173

1 copy

IDA

General W.Y. Smith, HQ	1 copy
Mr. Seymour Deitchman, HQ	1 copy
Mr. Philip Major, HQ	1 copy
Ms. Charlene Pandoli, HQ	1 copy
Dr. Jack Kramer, CSED	1 copy
Dr. Cathy Jo Linn, CSED	20 copies
Dr. Dennis W. Fife, CSED	50 copies
Ms. Julia Sensiba, CSED	2 copies
IDA Control & Distribution Vault	3 copies